ChatGPT

Multimodal User Profiling and Segmentation Methods

Understanding and segmenting users requires combining signals from text, vision, behavior, and contextual data. Below we survey techniques for each modality, methods for dynamic clustering, and emerging protocols for sharing user context across AI agents.

Text Analysis Techniques

Textual user data (messages, reviews, social posts) are analyzed via NLP to infer sentiment, personality, intent, and communication style. **Sentiment analysis** detects positive/negative/neutral tone using lexicon-based or machine-learning classifiers (e.g. fine-tuned BERT models) [1]. **Emotion detection** extends sentiment analysis to finer-grained emotions (joy, anger, etc.) [1]. **Personality profiling** (e.g. Big Five traits) can be estimated by mapping word usage or semantic features to psychometric traits (using tools like LIWC or deep learning on text embeddings). **Intent classification** (common in dialog systems) uses supervised models to map text to intent categories (often via transformer-based classifiers). **Communication style** (formality, verbosity, question vs assertive tone) can be captured by linguistic features (e.g. pronoun use, sentiment polarity shifts) or learned embeddings. In practice, deep learning models – such as CNNs or RNNs for text, or transformer networks – encode textual input and classify aspects like sentiment or persona. For example, transformer-based sentiment classifiers are state-of-the-art for many domains, and topic/intent models (LSTM or transformer) can profile user interests from posts.

- *Key methods:* Lexicon-based scoring (e.g. VADER), bag-of-words or TF-IDF with SVM; deep models (BERT/Transformer, CNN/RNN classifiers) for sentiment/intent; psycholinguistic tools (LIWC, Empath) or deep embeddings for personality traits.

Video and Visual Analysis

Video and image data provide rich nonverbal cues. **Facial expression recognition** uses computer vision to detect emotions from facial images. Modern systems use deep CNNs (e.g. variants of VGGFace, ResNet, or specialized models like DeepFace) to encode face images and classify expressions (happy, sad, angry, etc.) or continuous affect. **Affective computing** ("Emotion AI") analyzes human facial muscle patterns and body language to assess emotional state [2]. For example, deep convolutional neural networks trained on facial datasets can predict emotion labels; CNN-based "facial attribute" models (like IBM's DeepFace or FaceNet) serve as backbones for emotion inference [2] [3].

Gaze and eye-tracking: Computer vision can track eye gaze from ordinary video. By detecting eye/face landmarks and head pose, models estimate where the user is looking. Recent deep-learning methods use off-the-shelf cameras and neural networks (e.g. DeepLabCut) to map facial landmarks to gaze direction [4]. For instance, a shallow neural network trained on webcam frames can achieve <1° error in point-of-gaze [4]. This enables inference of user attention or interest in video interfaces.

Body posture and gesture analysis: Pose-estimation algorithms (OpenPose, MediaPipe) extract skeletal joints from video; machine learning models (CNN+RNN) then recognize gestures or emotional body language. For example, identifying crossed arms, gestures, or agitated movements can augment facial cues for mood analysis. While research on *body language analysis* is less standardized, emerging systems use skeleton-based CNN/RNN pipelines to classify gestures or detect stress levels in posture.

Visual sentiment: In addition to faces, overall visual context (scene, clothing, activity) can hint at user state. Multimodal vision-language models (e.g. CLIP-style embeddings) can relate user images or background scenes to mood or preferences. Some systems fuse audio (tone of voice) with video for richer affect recognition.

- *Key methods:* Face-detection + CNN classifiers for expression; CNN or attention models (e.g. CNN-LSTM) for video sequences; gaze tracking via landmark regression networks [4]; pose estimation + gesture classification models; multimodal fusion (e.g. combining audio/text from video with vision).

Behavioral and Biometric Data

Device and activity patterns: Logs of app usage, clickstreams, or sensor data reveal routines. Features like frequency/timing of app launches or website visits can be clustered into user "lifestyles" (e.g. night-owl vs early-bird, gaming vs productivity oriented). Time-series models (LSTM, ARIMA) capture daily or weekly cycles.

Keystroke and mouse dynamics: Users' typing rhythms and mouse movements carry identity and state. Keystroke-dynamics systems measure key-press durations and inter-key timings; machine learning (SVMs, neural nets) can identify users or infer stress from deviations in rhythm. Likewise, mouse-movement patterns (speed, trajectory) can profile behavior. Studies have shown that keyboard/mouse/touchscreen (KMT) dynamics can predict emotional arousal unobtrusively [5], because patterns like erratic typing or pausing correlate with frustration or attentiveness.

Wearable biometrics: Physiological signals (heart rate, skin conductance) from wearables enable emotional or health profiling. For example, **heart rate variability (HRV)** is a strong stress indicator (low HRV ⇔ high stress) [6]. Machine learning models (ensemble classifiers, deep nets) on HRV features can classify stress or relaxation states [6]. Sleep patterns (actigraphy, sleep-stage estimation) reveal chronic stress or mood (e.g. insomnia correlates with anxiety).

Gait and motion: Walking patterns are unique biometrics. Camera- or sensor-based gait recognition uses deep learning to identify individuals by how they walk [7]. Gait features can also hint at mood (e.g. slower gait in depression). Deep networks (CNNs, RNNs, attention models) have advanced gait analysis, extracting spatio-temporal features from video sequences [7].

Other biometric signals: Eye blink rate, facial temperature (thermal imaging), or EEG (brain waves) offer deeper state cues. These are more intrusive, but in lab settings, EEG + ML can even predict engagement or drowsiness.

- *Key methods:* Feature extraction from time-series and sensors (FFT, statistical features); supervised ML on sensor features; deep learning on raw physiological signals (CNNs on PPG waveforms, RNNs

on sequences). Federated learning and privacy-preserving analysis (differential privacy) are increasingly applied to biometric data to protect user privacy.

Temporal and Contextual Analysis

Users' behavior depends strongly on context (time, location, events). **Time-of-day and day-of-week:** Cyclic patterns (morning routines, weekend vs weekday) are modeled with calendar features or periodic autoencoders. Time-series forecasting (LSTM, Prophet) can learn when a user is active or receptive. For example, an agent may learn that a user checks social media every evening, so relevancy of notifications is higher at that time.

Location and mobility: GPS or WiFi logs reveal common places (home, work) and travel habits. Clustering location traces segments users into "commuters", "remote workers" etc. Spatio-temporal models (graph-based trajectories, HMMs) can predict next location or typical schedule. Location history also constrains content relevancy (e.g. recommending local events).

App and device usage: The set and timing of apps a user uses reflect context. Topic modeling or embedding of app categories over time can cluster usage patterns. For instance, frequent use of fitness apps correlates with health interest; spikes in finance app usage might indicate fiscal planning.

Social and calendar context: Social graph activity (who the user interacts with) and calendar events (meetings, vacations) modulate behavior. Context-aware recommender systems often include such features: e.g. if a high-priority meeting is on calendar, the user's priority for work-related alerts increases.

Context-aware modeling: Contextual factors are often encoded as features or latent variables. For example, an LSTM tag input sequences with time/location tokens; a context encoder outputs a "situation" embedding. Contextual bandit models use context features to rank content. In general, context-aware systems fuse sensor data, user preferences, and environmental cues to adapt behavior [8]. For instance, prior work notes that personal context like user schedule and preferences can markedly improve system personalization [8].

- *Key methods:* Time-series analysis (FFT, seasonal decomposition, LSTM), context fusion (concatenating time/location features into user embeddings), contextual multi-armed bandits for prioritization, and clustering users by combined feature vectors (time-of-use, geolocation).

Clustering and Segmentation with Dynamic Relevance

To segment users, features from all modalities can be combined into multimodal embeddings. **Multimodal clustering** extracts shared representations across data types and groups similar users [9]. For example, autoencoders or deep multimodal networks learn a joint latent space for text, image, and behavioral features, then cluster (e.g. with k-means or Gaussian Mixture Models) in that space [9]. Graph-based clustering (using Graph Neural Networks) can incorporate social or device-relations: users are nodes, edges represent interactions or similarity, and GCNs can produce node embeddings which are then clustered.

Often segmentation is *hierarchical*: a top-level cluster captures coarse personality or lifestyle, and finer sub-clusters capture situational segments. **Tiered relevancy** means ranking clusters per context. For example, a user might generally belong to "tech-savvy" and "fitness enthusiast" clusters, but an agent "zooms in" on

the fitness cluster when suggesting a running playlist in the morning. This dynamic shifting can be handled by context-gated clustering: weighting feature subsets or applying attention over cluster centroids based on current context.

- *Key methods:* Unsupervised clustering (k-means, DBSCAN, hierarchical, spectral clustering) on fused feature vectors; mixture models that allow soft cluster membership; manifold learning (t-SNE, UMAP) for visualization of segments; dynamic/fuzzy clustering where membership probabilities change with context. **Graph Neural Networks (GNNs)** have been used to capture relational patterns and recompute clusters as edges change. Recent surveys note that DL-based multimodal clustering (using CNNs, RNNs, GCNs) is an active area [10].

Predictive Modeling Approaches

Once users are modeled, predictive algorithms rank content or anticipate behavior. **Deep learning** is widely used: multi-layer perceptrons or CNNs for static features; LSTM/GRU/Transformer for sequences (e.g. predicting next action in a clickstream or estimating mood from temporal signals) [11]. For instance, bidirectional LSTM models have been trained on user post histories to classify topical interests, forming evolving user profiles [11]. **Graph Neural Networks** allow leveraging relational data: e.g. representing users in a graph (with edges for friendships or co-occurrence of interests) and using GCNs or GraphSAGE to propagate influence and predict preferences. GNNs can also aggregate multimodal signals: one can construct a heterogeneous graph with nodes for users, content, locations, and apply graph convolution to rank relevance.

Time-series forecasting and anomaly detection: RNNs or temporal convolutional networks (TCNs) model user metrics (daily steps, mood scores) to predict trends or detect regime changes. Techniques like Transformers (e.g. Informer, Temporal Fusion) are also used for long-range time dependencies.

Attention and contextual models: Newer models use attention mechanisms to weigh which modalities or past events are most relevant to a prediction. For instance, an agent might use a transformer encoder on the sequence of user activities, with attention focusing on recent actions or on those matching the current query.

Ensemble and hybrid models: Practical systems often combine methods: e.g. feed RNN outputs into a GNN, or use CNNs on images and then combine embeddings with text using a fusion network. In summary, prediction typically leverages deep neural architectures (CNN, RNN, Transformer, GNN) trained on rich feature sets. A recent review notes that **Bi-LSTM and GRU networks** effectively capture both short-term and long-term patterns in user activity sequences to dynamically update profiles [11].

Example Segmentation and Modeling Summary

Approach	Use Case	Data / Model
User Clustering	Marketing personas, targeting	Multi-view clustering (k-means, GMM on text+usage features) [9]
Graph-based Segmentation	Social network analysis	GNN (GCN/GraphSAGE) on user-item or social graph

Approach	Use Case	Data / Model
Fuzzy/Hierarchical	Flexible profiles, tiered interests	Fuzzy C-means; hierarchical clustering tree
Sequence Modeling	Predict next action/mood	LSTM/GRU/Transformer on time-series features [11]
Multimodal Fusion	Combined vision+audio/ text profiling	CNN+RNN fusion, attention models

(Table: Representative segmentation and predictive methods by goal.)

Context-Sharing Protocol Architectures

To share user context among AI agents, emerging standardized protocols aim for interoperability and decentralized design. For example, **Google's Agent-to-Agent (A2A) Protocol** provides a common language for agents to exchange goals and context [12]. A2A messages use structured JSON-LD "Agent Cards" so that any A2A-compliant agent can understand context and delegate tasks [12]. Similarly, **Anthropic's Model Context Protocol (MCP)** and related efforts define JSON-RPC or HTTP interfaces so tools (data sources, services) can be plugged into agents with shared context [13] [14]. MCP enforces well-defined inputs/outputs and a shared context object so that any MCP tool or agent understands the user's current state [13] [15].

A recent survey identifies four key protocols: MCP, **Agent Communication Protocol (ACP)**, **A2A**, and **Agent Network Protocol (ANP)** [16]. MCP (as above) focuses on tool invocation via JSON-RPC; ACP adds RESTful multipart messaging for multimodal responses; A2A enables peer-to-peer task outsourcing among agents; ANP uses decentralized identifiers (DIDs) and linked data (JSON-LD) for open discovery of agents [16]. Together, these aim to form a layered architecture: MCP for context-rich queries, ACP for complex messaging, A2A for collaborative workflows, and ANP for a decentralized agent marketplace [16].

Architecturally, such protocols emphasize **lightweight context** exchange. Rather than each agent profiling users independently, they publish small **context tokens** or references. For example, under MCP an agent might query a context server for a user's profile summary instead of recalculating it locally [13]. Decentralized storage (peer-to-peer or federated) is envisioned: user context can reside on the user's device or personal pod (like Solid), and agents fetch only needed attributes via secure APIs. Some proposals use distributed ledgers or DIDs (as in ANP) to index context capabilities without central servers, enhancing privacy and resilience [16].

Example frameworks: Early agent standards (FIPA-ACL, KQML) laid groundwork for agent messaging, but modern efforts (MCP/A2A/ACP/ANP) explicitly target LLM-powered assistants. IBM and community projects provide SDKs for MCP; Google and open-source groups are defining A2A schemas. Other related ideas include shared knowledge graphs or ontologies for user context, and federated learning protocols where agents learn from shared anonymized user data.

Protocol / Framework	Primary Focus	Key Features
Model Context Protocol (MCP)	Standardizing agent-tool communication [13]	JSON-RPC/HTTP interface; modular tools; shared "context" object; fosters reuse [13]
Agent-to-Agent (A2A)	Cross-agent collaboration [12]	Structured "Agent Cards" with goals and context; vendor-neutral; peer interactions [12]
Agent Communication Protocol (ACP)	Multi-modal messaging (not yet ratified)	RESTful requests with asynchronous streams; designed for attachments/audio/video
Agent Network Protocol (ANP)	Decentralized agent marketplace [16]	Uses decentralized identifiers (DIDs) and JSON-LD graphs for discovery [16] ; supports open agent networks
Other Standards	–	Existing standards like ActivityPub (social sharing) or Personal Data Stores (Solid) for user data

(Table: Emerging context-sharing protocols for AI agents; see [16] [13] .)

Security, Privacy and Efficiency

Protocols and models must safeguard user data. **Privacy** is addressed via data minimization and federated learning: agents can perform on-device user profiling and share only abstracted context (e.g. "shopping preference: sports apparel" instead of raw purchase history). **Encryption and Access Control:** communications (MCP, A2A messages) are expected to use TLS and signed tokens; decentralized IDs (DID) and VC (verifiable credentials) schemes (as in ANP) ensure only authorized agents see a user's context [16] . **Differential privacy** or secure enclaves may be applied when aggregating user data (e.g. computing a community preference from private logs).

For **efficiency**, context messages are kept lightweight (metadata, pointers, or compressed embeddings) to avoid transferring raw profiles. Agents reuse shared context servers or vector stores to avoid redundant computation. Protocols like MCP support both local subprocess tools (stdin/stdout) and scalable SSE/HTTP tools [17] , allowing quick local contexts or remote lookups as needed.

In summary, multi-agent systems aim for **decentralized, interoperable context sharing**: users retain control (via personal data pods or on-device models), and agents exchange only what's necessary via standardized schemas. Emerging architectures emphasize security (encrypted channels, authenticated endpoints), privacy (principle of least privilege, on-device learning), and efficiency (modular tool invocation, caching) to build scalable, trustable AI ecosystems.

Sources: Contemporary surveys and blog posts detail many of the above methods. For multimodal clustering and deep learning integration, see Raya et al. [9] . Computer vision methods for facial and emotion analysis are reviewed in emotion AI literature [18] [2] . Biometric profiling (keystroke, HRV, gait) is discussed in recent studies [5] [6] [7] . For context protocols, refer to the MCP introduction [13] and the agent interoperability survey [16] .

1 A review on sentiment analysis and emotion detection from text - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC8402961/

2 3 18 AI Emotion Recognition and Sentiment Analysis - viso.ai
https://viso.ai/deep-learning/visual-emotion-ai-recognition/

4 Frontiers | A Deep Learning-Based Approach to Video-Based Eye Tracking for Human Psychophysics
https://www.frontiersin.org/journals/human-neuroscience/articles/10.3389/fnhum.2021.685830/full

5 researchportal.northumbria.ac.uk
https://researchportal.northumbria.ac.uk/files/65202879/Final_published_version.pdf

6 Stress management with HRV following AI, semantic ontology, genetic algorithm and tree explainer | Scientific Reports
https://www.nature.com/articles/s41598-025-87510-w?error=cookies_not_supported&code=a7f62d49-04b2-4f8c-9c61-a36950721736

7 Emerging trends in gait recognition based on deep learning: a survey - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC11323174/

8 Context-aware systems: A literature review and classification
https://ics.uci.edu/~wmt/courses/Inf241S14/papers/science.pdf

9 10 Multi-modal data clustering using deep learning: A systematic review | CoLab
https://colab.ws/articles/10.1016%2Fj.neucom.2024.128348

11 Temporal dynamics of user activities: deep learning strategies and mathematical modeling for long-term and short-term profiling | Scientific Reports
https://www.nature.com/articles/s41598-024-64120-6?error=cookies_not_supported&code=aff75138-d4c8-4ffd-b0de-cf8401c7ebf0

12 Google’s A2A Protocol: A New Standard for AI Agent Interoperability
https://www.vktr.com/ai-market/googles-a2a-protocol-a-new-standard-for-ai-agent-interoperability/

13 15 17 Decentralized Tools in AI Agents Using Model Context Protocol (MCP)
https://community.ibm.com/community/user/blogs/anshad-mohamed/2025/05/05/mcp

14 Advancing Multi-Agent Systems Through Model Context Protocol: Architecture, Implementation, and Applications
https://arxiv.org/html/2504.21030v1

16 A Survey of Agent Interoperability Protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP)
https://arxiv.org/html/2505.02279v1