Algorithm 1 Convert Policies Into Embeddings

```
Initialize empty dictionary policy_embs (key=policy, value=embedding)
for each party in data do
    for each policy in data[party] do
        Format policy into agree and disagree statements via LLM
        Convert agree and disagree statements into embeddings
        Compute delta_emb as agree minus disagree
        Add delta_emb to policy_embs
    end for
end for
Store policy_embs as result
```

Algorithm 2 Convert Parties Into Embeddings

```
Load policy_embs
Initialize empty dictionary party_embs (key=party, value=embeddings)
for each party in data do
    for each policy in policy_embs do
        if data[party][policy] exists then
            Append policy_embs[policy] to policy[party]
        end if
    end for
end for
for each party in party_embs do
    Set party_embs[policy] to average party_embs[policy]
end for
Store party_embs as result
```

Algorithm 3 Find Closest Political Party Based on User Policy Preferences

Input: JSON file `emb_data.json` with party and policy embeddings
Output: Sorted list of parties by proximity to user embedding
Load JSON data from `emb_data.json`
Decode party embeddings into dictionary `parties`
Decode policy embeddings into dictionary `policies`
Initialize user embedding `user_emb` as zero vector (same shape as policy embeddings)
Initialize counter `n` $\leftarrow 1$
for each policy in `policies` **do**
    Prompt user for preference score `score` $\in [-1, 1]$
    Update `user_emb` $\leftarrow$ `user_emb` $+$ `score` $\times$ `policies[policy]`
    Increment `n` $\leftarrow$ `n` $+ 1$
    Compute distances: `1 - dot(user_emb / n, party_emb)` for each party
    Sort parties by ascending distance
    Output sorted party list
end for