In many intent-centric applications - especially those focused on demand-side aggregation - the “hard problem” (or so to speak) to solve is that of fuzzy matching

and requirement synthesis

:

- Fuzzy matching

, i.e. how to recognize that user intents with descriptions of requirements for a product to be produced or an outcome to be sought written in natural language are similar to each other, and

- Requirement synthesis

, i.e. how to combine two “similar enough” user intents in a way that preserves the essential requirements of each intent being so combined.

These problems are not trivial - in full generality, they are impossible, since this would require a perfect model of the world - so we will not attempt to “solve” them, per se, but rather clarify the interfaces and point out possible decompositions into sub-problems and ways to introduce humans (or LLMs, or other entities with world-models) in the loop to help with synthesis.

I will use Public Signal as a prototypical example to explain what I mean concretely here. In Public Signal, users submit intents which describe a willingness to pay for a particular product, if only it existed. For example (using some pseudo-syntax), imagine that we have intents as follows:

- Bob is willing to pay up to 100 USD for a simple smartphone with a web browser, hardware kill switches, at least one SIM slot, and open-source hardware/software stack. It must weigh under 500 grams and have at least a day’s worth of battery life in regular use.
- Sally is willing to pay up to 150 USD for a simple smartphone with a web browser, two SIM slots, and an open-source hardware/software stack. It must weigh under 750 grams and have at least a day’s worth of battery life in regular use.
- Charlie is willing to pay up to 75 USD for a simple smartphone with a web browser and two SIM slots. It must have at least two days worth of battery life.

Reading this text, one can straightforwardly infer classes of potential products which would satisfy different combinations of these intents. What is not clear is how best to represent these kinds of requirements in a way which can be deterministically processed - since in reading this text and synthesizing possible combinations, one uses an understanding of the world - specifically smartphones - to understand which entities are unique in the description of requirements and how they match up. I can imagine a few ways the application design might go:

1. Users of the application might construct and share specific product ontologies

which define standardized features, attributes, and synthesis rules for a particular category of products. For example, if “smartphone” is a category, “weight” and “battery life” might be standardized attributes. Intents can then either be written using these specific ontologies directly, or perhaps converted from natural-language descriptions using LLMs (and then displayed to the user for confirmation). Solvers (and potential producers of the product(s) in question) can use the synthesis rules to determine how requirements can be combined.

This option is simple, straightforward, and safe, but it puts a lot of onus on the authors of these product ontologies to capture salient features and attributes, and we will need to think more about how these ontologies are developed, shared, updated, etc. - and how disputes are handled after the products are actually built (as to whether or not they matched the requirements specified in the ontology).

1. We might focus exclusively on the problem of composition of natural language descriptions, and aim for Public Signal to ultimately produce some composite intent with a natural language description of the product to be produced. The question here then is how these natural language descriptions can be combined. Users of Public Signal could choose a designated agent - perhaps a sufficiently neutral party or an LLM - who then has a special permission to combine intents and produce composed natural language descriptions which they assert will satisfy the originally articulated requirements. In the limit case, there can be many such parties at different stages of composition, and users could delegate this “right of composition” along specific lines of trust and expertise (e.g. I might delegate mine to a smartphone hardware expert friend).

This approach is much more general and can adopt a topology which requires much less agreement on natural language ontologies, but requires a lot of online user interaction - perhaps workable for a slower-paced application such as Public Signal - but we need to think through the specifics. There’s also some “compositional MEV” here that will require analysis.

1. Some combination of (1) and (2), possibly with additional incentives such as staking on the accuracy of compositions, where future post-production disputes could lead to slashing.

I’ll leave my thoughts here for now. This problem is quite general and also very relevant to topics such as governance. Inviting opinions by @apriori @degregat @nikete especially.