1. Monitor the services mentioned above for new blips. For each blip, tag it with user and location and store it in their system. Given that these services are interconnected, they want to have de-duplication in place (for instance, a tweet and a status update with the same text and from the same person should count as one blip and only be stored once). They do not care if they do it in real time or later on, but eventually (say, one day after the blips were first encountered) they do not want any duplicates in their store. Depending on the service, blips may come already tagged with a value (e.g., hash tags in tweets); these tags should be isolated and stored by themselves, but without losing the connection to the blip.

2. In addition to explicit tags, there should also be implicit tags, i.e. tags one can identify by performing some post-processing of the blip. This need not necessarily be instant – they can afford to do it in bulk, though latency should not be more than a couple of hours after the blip was first encountered.

3. They want to be able to identify networks of users. For instance, users who tend to respond to one another through tweets are most likely in the same social cycle. Same thing applies to common tags: users using common tags repeatedly, most likely share similar interests.

4. They should be able to aggregate among multiple dimensions. For instance, they would like to group tags by location, or locations by tags. Given the volume of data, such computations are to be performed in bulk. However, the results should be stored in an efficient way that allows for fast retrieval, so they can provide their service.

5. They would also like to be able to detect trends, but ideally they would also like to have a notion of tag disambiguation. They should be able to detect whether a blip refers to a product, or an event, or a movie, or a TV series, etc. The way they envisage doing this is by identifying certain information in the blip. For instance, the use of watched will most likely refer to a movie, or a TV series.

6. What they effectively want to do is to send a message to each user after they detect a blip from him/her to alert him of related information. An example might be a user interested in theatre: they should be able to provide her/him with recommendations for current theatre performances in her/his area that she/he might want to see. This can be done either in real-time (for instance, using a direct blip to the user) or in bulk (for instance, an overnight email with a list of recommendations).

7. Recommendations might be in the form of products/events or in the form of other users.

8. In addition to the locality aspects of the service, there should be a temporal aspect as well. Recently used tags should have more importance than tags used a few months ago, even if the frequency of earlier tags is higher.

9. The system should be always running and it should be elastic, in the sense that it should be able to allocate resources dynamically to the different types of computation. For instance, if there is a high load of blips coming in, it is more important to capture the stream of blips as opposed to perfoming aggregation on the background. Likewise, if the system is low on load, it should be able to scale down, either by reallocating resources to background processes, or even freeing them up completely.