# Sims Candidate Source

Offers various online sources for finding similar accounts based on a given user, whether it is the target user or an account candidate.

## Sims

The objective is to identify a list of K users who are similar to a given user. In this scenario, we primarily focus on finding similar users as "producers" rather than "consumers." Sims has two steps: candidate generation and ranking.

### Sims Candidate Generation

With over 700 million users to consider, there are multiple ways to define similarities. Currently, we have three candidate sources for Sims:

\*\*CosineFollow\*\* (based on user-user follow graph): The similarity between two users is defined as the cosine similarity between their followers. Despite sounding simple, computing all-pair similarity on the entire follow graph is computationally challenging. We are currently using the WHIMP algorithm to find the top 1000 similar users for each user ID. This candidate source has the largest coverage, as it can find similar user candidates for more than 700 million users.

\*\*CosineList\*\* (based on user-list membership graph): The similarity between two users is defined as the cosine similarity between the lists they are included as members (e.g., [here](https://twitter.com/jack/lists/memberships) are the lists that @jack is on). The same algorithm as CosineFollow is used.

\*\*Follow2Vec\*\* (essentially Word2Vec on user-user follow graph): We first train the Word2Vec model on follow sequence data to obtain users' embeddings and then find the most similar users based on the similarity of the embeddings. However, we need enough data for each user to learn a meaningful embedding for them, so we can only obtain embeddings for the top 10 million users (currently in production, testing 30 million users). Furthermore, Word2Vec model training is limited by memory and computation as it is trained on a single machine.

##### Cosine Similarity

A crucial component in Sims is calculating cosine similarities between users based on a user-X (X can be a user, list, or other entities) bipartite graph. This problem is technically challenging and took several years of effort to solve.

The current implementation uses the algorithm proposed in [When hashes met wedges: A distributed algorithm for finding high similarity vectors. WWW 2017](https://arxiv.org/pdf/1703.01054.pdf)

### Sims Ranking

After the candidate generation step, we can obtain dozens to hundreds of similar user candidates for each user. However, since these candidates come from different algorithms, we need a way to rank them. To do this, we collect user feedback.

We use the "Profile Sidebar Impressions & Follow" (a module with follow suggestions displayed when a user visits a profile page and scrolls down) to collect training data. To alleviate any system bias, we use 4% of traffic to show randomly shuffled candidates to users and collect positive (followed impression) and negative (impression only) data from this traffic. This data is used as an evaluation set. We use a portion of the remaining 96% of traffic for training data, filtering only for sets of impressions that had at least one follow, ensuring that the user taking action was paying attention to the impressions.

The examples are in the format of (profile\_user, candidate\_user, label). We add features for profile\_users and candidate\_users based on some high-level aggregated statistics in a feature dataset provided by the Customer Journey team, as well as features that represent the similarity between the profile\_user and candidate\_user.

We employ a multi-tower MLP model and optimize the logistic loss. The model is refreshed weekly using an ML workflow.

We recompute the candidates and rank them daily. The ranked results are published to the Manhattan dataset.