# SimClusters ANN

SimClusters ANN is a service that returns tweet candidate recommendations given a SimClusters embedding. The service implements tweet recommendations based on the Approximate Cosine Similarity algorithm.

The cosine similarity between two Tweet SimClusters Embedding represents the relevance level of two tweets in SimCluster space. The traditional algorithm for calculating cosine similarity is expensive and hard to support by the existing infrastructure. Therefore, the Approximate Cosine Similarity algorithm is introduced to save response time by reducing I/O operations.

## Background

SimClusters V2 runtime infra introduces the SimClusters and its online and offline approaches. A heron job builds the mapping between SimClusters and Tweets. The job saves top 400 Tweets for a SimClusters and top 100 SimClusters for a Tweet. Favorite score and follow score are two types of tweet score. In the document, the top 100 SimClusters based on the favorite score for a Tweet stands for the Tweet SimClusters Embedding.

The cosine similarity between two Tweet SimClusters Embedding presents the relevant level of two tweets in SimCluster space. The score varies from 0 to 1. The high cosine similarity score(>= 0.7 in Prod) means that the users who like two tweets share the same SimClusters.

SimClusters from the Linear Algebra Perspective discussed the difference between the dot-product and cosine similarity in SimCluster space. We believe the cosine similarity approach is better because it avoids the bias of tweet popularity.

However, calculating the cosine similarity between two Tweets is pretty expensive in Tweet candidate generation. In TWISTLY, we scan at most 15,000 (6 source tweets \* 25 clusters \* 100 tweets per clusters) tweet candidates for every Home Timeline request. The traditional algorithm needs to make API calls to fetch 15,000 tweet SimCluster embeddings. Consider that we need to process over 6,000 RPS, it’s hard to support by the existing infrastructure.

## SimClusters Approximate Cosine Similarity Core Algorithm

1. Provide a source SimCluster Embedding \*SV\*, \*SV = [(SC1, Score), (SC2, Score), (SC3, Score) …]\*

2. Fetch top \*M\* tweets for each Top \*N\* SimClusters based on SV. In Prod, \*M = 400\*, \*N = 50\*. Tweets may appear in multiple SimClusters.

| | | | |

|---|---|---|---|

| SC1 | T1:Score | T2: Score | ... |

| SC2 | T3: Score | T4: Score | ... |

3. Based on the previous table, generate an \*(M x N) x N\* Matrix \*R\*. The \*R\* represents the approximate SimCluster embeddings for \*MxN\* tweets. The embedding only contains top \*N\* SimClusters from \*SV\*. Only top \*M\* tweets from each SimCluster have the score. Others are 0.

| | SC1 | SC2 | ... |

|---|---|---|---|

| T1 | Score | 0 | ... |

| T2 | Score | 0 | ... |

| T3 | 0 | Score | ... |

4. Compute the dot product between source vector and the approximate vectors for each tweet. (Calculate \*R • SV^T\*). Take top \*X\* tweets. In Prod, \*X = 200\*

5. Fetch \*X\* tweet SimClusters Embedding, Calculate Cosine Similarity between \*X\* tweets and \*SV\*, Return top \*Y\* above a certain threshold \*Z\*.

Approximate Cosine Similarity is an approximate algorithm. Instead of fetching \*M \* N\* tweets embedding, it only fetches \*X\* tweets embedding. In prod, \*X / M \* N \* 100% = 6%\*. Based on the metrics during TWISTLY development, most of the response time is consumed by I/O operation. The Approximate Cosine Similarity is a good approach to save a large amount of response time.

The idea of the approximate algorithm is based on the assumption that the higher dot-product between source tweets’ SimCluster embedding and candidate tweet’s limited SimCluster Embedding, the possibility that these two tweets are relevant is higher. Additional Cosine Similarity filter is to guarantee that the results are not affected by popularity bias.

Adjusting the M, N, X, Y, Z is able to balance the precision and recall for different products. The implementation of approximate cosine similarity is used by TWISTLY, Interest-based tweet recommendation, Similar Tweet in RUX, and Author based recommendation. This algorithm is also suitable for future user or entity recommendation based on SimClusters Embedding.

# -------------------------------

# Build and Test

# -------------------------------

Compile the service

$ ./bazel build simclusters-ann/server:bin

Unit tests

$ ./bazel test simclusters-ann/server:bin

# -------------------------------

# Deploy

# -------------------------------

## Prerequisite for devel deployments

First of all, you need to generate Service to Service certificates for use while developing locally. This only needs to be done ONCE:

To add cert files to Aurora (if you want to deploy to DEVEL):

```

$ developer-cert-util --env devel --job simclusters-ann

```

## Deploying to devel/staging from a local build

Reference -

$ ./simclusters-ann/bin/deploy.sh --help

Use the script to build the service in your local branch, upload it to packer and deploy in devel aurora:

$ ./simclusters-ann/bin/deploy.sh atla $USER devel simclusters-ann

You can also deploy to staging with this script. E.g. to deploy to instance 1:

$ ./simclusters-ann/bin/deploy.sh atla simclusters-ann staging simclusters-ann <instance-number>

## Deploying to production

Production deploys should be managed by Workflows.

\_Do not\_ deploy to production unless it is an emergency and you have approval from oncall.

##### It is not recommended to deploy from Command Lines into production environments, unless 1) you're testing a small change in Canary shard [0,9]. 2) Tt is an absolute emergency. Be sure to make oncalls aware of the changes you're deploying.

$ ./simclusters-ann/bin/deploy.sh atla simclusters-ann prod simclusters-ann <instance-number>

In the case of multiple instances,

$ ./simclusters-ann/bin/deploy.sh atla simclusters-ann prod simclusters-ann <instance-number-start>-<instance-number-end>

## Checking Deployed Version and Rolling Back

Wherever possible, roll back using Workflows by finding an earlier good version and clicking the "rollback" button in the UI. This is the safest and least error-prone method.