\*\*\*\*\*\*\*\*

Overview

\*\*\*\*\*\*\*\*

This job reads embedding data from HDFS in the embedding formats supported by the cortex MLX team. It converts that data into the ANN format and adds it to an ANN index. The ANN index is serialized and save to disk.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Running In Aurora

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Set up example

==============

This job builds an ANN index based on hnsw algorithm using user embeddings available in hdfs.

.. code-block:: bash

$ export JOB\_NAME=ann\_index\_builder

$ export OUTPUT\_PATH=hdfs:///user/$USER/${JOB\_NAME}\_test

$ CPU=32 RAM\_GB=150 DISK\_GB=60 aurora job create smf1/$USER/devel/$JOB\_NAME ann/src/main/aurora/index\_builder/aurora\_builder.aurora \

--bind=profile.name=$JOB\_NAME \

--bind=profile.role=$USER \

--bind=profile.output\_dir=$OUTPUT\_PATH \

--bind=profile.entity\_kind=user \

--bind=profile.embedding\_args='--input.embedding\_format tab --input.embedding\_path /user/cortex-mlx/official\_examples/ann/non\_pii\_random\_user\_embeddings\_tab\_format' \

--bind=profile.num\_dimensions=300 \

--bind=profile.algo=hnsw \

--bind=profile.ef\_construction=200 \

--bind=profile.max\_m=16 \

--bind=profile.expected\_elements=10000000 \

--bind=profile.metric=InnerProduct \

--bind=profile.concurrency\_level=32 \

--bind=profile.hadoop\_cluster=dw2-smf1

This job builds an ANN index based on hnsw algorithm using producer embeddings (Major version 1546473691) available in feature store.

.. code-block:: bash

$ export JOB\_NAME=ann\_index\_builder

$ export OUTPUT\_PATH=hdfs:///user/$USER/${JOB\_NAME}\_test

$ CPU=32 RAM\_GB=150 DISK\_GB=60 aurora job create smf1/$USER/devel/$JOB\_NAME ann/src/main/aurora/index\_builder/aurora\_builder.aurora \

--bind=profile.name=$JOB\_NAME \

--bind=profile.role=$USER \

--bind=profile.output\_dir=$OUTPUT\_PATH \

--bind=profile.entity\_kind=user \

--bind=profile.embedding\_args='--input.feature\_store\_embedding ProducerFollowEmbedding300Dataset --input.feature\_store\_major\_version 1546473691 --input.date\_range 2019-01-02' \

--bind=profile.num\_dimensions=300 \

--bind=profile.algo=hnsw \

--bind=profile.ef\_construction=200 \

--bind=profile.max\_m=16 \

--bind=profile.expected\_elements=10000000 \

--bind=profile.metric=InnerProduct \

--bind=profile.concurrency\_level=32 \

--bind=profile.hadoop\_cluster=dw2-smf1

\*\*\*\*\*\*\*\*\*\*\*\*\*

Job arguments

\*\*\*\*\*\*\*\*\*\*\*\*\*

Enviroment variables (resources):

==============

- \*\*CPU\*\* Number of cpu cores (default: 32)

- \*\*RAM\_GB\*\* RAM in gigabytes (default: 150)

- \*\*DISK\_GB\*\* Disk in gigabytes (default: 60)

General arguments (specified as \*\*--profile.{options}\*\*):

==============

- \*\*name\*\* Aurora job name

- \*\*role\*\* Aurora role

- \*\*hadoop\_cluster\*\* Hadoop cluster for data. dw2-smf1/proc-atla.

- \*\*input\_dir\*\* Path of saved embeddings in hdfs without prefixing `hdfs://`

- \*\*entity\_kind\*\* The type of entity id that is use with the embeddings. Possible options:

- word

- url

- user

- tweet

- tfwId

- \*\*embedding\_args\*\* Embedding format args. See the documentation in `com.twitter.cortex.ml.embeddings.common.EmbeddingFormatArgsParser` for a full explanation of the input options. Possible options:

1. \*\*input.embedding\_format\*\* Format of the serialized embedding.

- usertensor

- usercontinuous

- comma

- tab

2. \*\*input.embedding\_path\*\* Path of saved embeddings in hdfs without prefixing `hdfs://`

3. \*\*input.{feature\_store\_args}\*\* For feature store related args like `feature\_store\_embedding`, `feature\_store\_major\_version`, `date\_range`:

- \*\*output\_dir\*\* Where to save the produced serialized ann index. Save to HDFS by specifying the full URI. e.g `hdfs://hadoop-dw2-nn.smf1.twitter.com/user/<user>/index\_file` or using the default cluster `hdfs:///user/<user>/index\_file`.

- \*\*num\_dimensions\*\* Dimension of embedding in the input data. An exception will be thrown if any entry does not have a number of dimensions equal to this number.

- \*\*metric\*\* Distance metric (InnerProduct/Cosine/L2)

- \*\*concurrency\_level\*\* Specifies how many parallel inserts happen to the index. This should probably be set to the number of cores on the machine.

- \*\*algo\*\* The kind of index you want to ouput. The supported options right now are:

1. \*\*hnsw\*\* (Metric supported: Cosine, L2, InnerProduct)

.. \_hnsw: https://arxiv.org/abs/1603.09320

- \*\*ef\\_construction\*\* : Larger value increases build time but will give better recall. Good start value : 200

- \*\*max\\_m\*\* : Larger value increases will increase the index size but will give better recall. Optimal Range : 6-48. Good starting value 16.

- \*\*expected\\_elements\*\* : Approximate number of elements that will be indexed.

2. \*\*annoy\*\* (Metric supported: Cosine, L2)

.. \_annoy: https://github.com/spotify/annoy

- \*\*annoy\\_num\\_trees\*\* This parameter is required for annoy. From the annoy documentation: num\_trees is provided during build time and affects the build time and the index size. A larger value will give more accurate results, but larger indexes.

3. \*\*brute\_force\*\* (Metric supported: Cosine, L2, InnerProduct)

Developing locally

===================

For building and testing custom ann index builder job,

You can create job bundle locally, upload to packer and then it can be used with the job using `profile.packer\_package` for name, `profile.packer\_role` for role and `profile.packer\_version` for bundle version.

.. code-block:: bash

./bazel bundle ann/src/main/scala/com/twitter/ann/scalding/offline/indexbuilder:indexbuilder-deploy \

--bundle-jvm-archive=zip

.. code-block:: bash

packer add\_version --cluster=atla <role> <package\_name> dist/indexbuilder-deploy.zip