# Earlybird Light Ranker

\*Note: the light ranker is an old part of the stack which we are currently in the process of replacing.

The current model was last trained several years ago, and uses some very strange features.

We are working on training a new model, and eventually rebuilding this part of the stack entirely.\*

The Earlybird light ranker is a logistic regression model which predicts the likelihood that the user will engage with a

tweet.

It is intended to be a simplified version of the heavy ranker which can run on a greater amount of tweets.

There are currently 2 main light ranker models in use: one for ranking in network tweets (`recap\_earlybird`), and

another for

out of network (UTEG) tweets (`rectweet\_earlybird`). Both models are trained using the `train.py` script which is

included in this directory. They differ mainly in the set of features

used by the model.

The in network model uses

the `src/python/twitter/deepbird/projects/timelines/configs/recap/feature\_config.py` file to define the

feature configuration, while the

out of network model uses `src/python/twitter/deepbird/projects/timelines/configs/rectweet\_earlybird/feature\_config.py`.

The `train.py` script is essentially a series of hooks provided to for Twitter's `twml` framework to execute,

which is included under `twml/`.

### Features

The light ranker features pipeline is as follows:

![earlybird\_features.png](earlybird\_features.png)

Some of these components are explained below:

- Index Ingester: an indexing pipeline that handles the tweets as they are generated. This is the main input of

Earlybird, it produces Tweet Data (the basic information about the tweet, the text, the urls, media entities, facets,

etc) and Static Features (the features you can compute directly from a tweet right now, like whether it has URL, has

Cards, has quotes, etc); All information computed here are stored in index and flushed as each realtime index segments

become full. They are loaded back later from disk when Earlybird restarts. Note that the features may be computed in a

non-trivial way (like deciding the value of hasUrl), they could be computed and combined from some more "raw"

information in the tweet and from other services.

Signal Ingester: the ingester for Realtime Features, per-tweet features that can change after the tweet has been

indexed, mostly social engagements like retweetCount, favCount, replyCount, etc, along with some (future) spam signals

that's computed with later activities. These were collected and computed in a Heron topology by processing multiple

event streams and can be extended to support more features.

- User Table Features is another set of features per user. They are from User Table Updater, a different input that

processes a stream written by our user service. It's used to store sparse realtime user

information. These per-user features are propagated to the tweet being scored by

looking up the author of the tweet.

- Search Context Features are basically the information of current searcher, like their UI language, their own

produced/consumed language, and the current time (implied). They are combined with Tweet Data to compute some of the

features used in scoring.

The scoring function in Earlybird uses both static and realtime features. Examples of static features used are:

- Whether the tweet is a retweet

- Whether the tweet contains a link

- Whether this tweet has any trend words at ingestion time

- Whether the tweet is a reply

- A score for the static quality of the text, computed in TweetTextScorer.java in the Ingester. Based on the factors

such as offensiveness, content entropy, "shout" score, length, and readability.

- tweepcred, see top-level README.md

Examples of realtime features used are:

- Number of tweet likes/replies/retweets

- pToxicity and pBlock scores provided by health models