Making Machine Learning Easy with Embeddings

Dan Shiebler
Twitter Cortex
dshiebler@twitter.com

Abhishek Tayal
Twitter Cortex
atayal@twitter.com

ABSTRACT

Modeling teams at Twitter face a variety of uniquely hard, yet fundamentally related machine learning problems. For example, tasks as different as ad serving, abuse detection and user timeline construction all rely on powerful representations of user and content entities. In addition, because of Twitter's realtime nature, entity data distributions are constantly in flux, so these representations must be frequently updated. By generating high quality, up-to-date representations and sharing them broadly across teams, we can decrease duplication of efforts and multiplicatively increase cross-team modeling productivity. At Twitter Cortex, we are making these representations "first class citizens" of the Twitter ML platform by commoditizing tools and pipelines that create high quality, custom, regularly retrained, benchmarked and centrally hosted embeddings.

Introduction

Machine Learning systems need to operate on many types of entities, such as images, text, audio, and music. But most Machine Learning algorithms only understand one kind of input - a vector. Therefore a critical component of any Machine Learning workflow is the process of converting entities to vector representations, or featurization. A common way to featurize an entity is to use a predefined algorithm that extracts a vector representation, such as a Bag of Words model. An alternative technique which often shows better results is to use learned features, also known as embeddings. Since a single learned embedding is often useful for a variety of tasks, sharing embeddings between teams that work with similar data reduces modeling effort and improves results. At Twitter, we have achieved excellent results by applying embeddings to problems like email recommendation, user timeline construction and onboarding.

However, unlike hand crafted features, which are generated by rule based algorithms, learned embeddings are themselves outputs of Machine Learning models that require data to train and may be cumbersome to store and deploy. Like the models that they are used with, embeddings need to be regularly retrained and benchmarked - especially in a constantly changing system like Twitter. In order to address these issues and reap the benefits of embeddings, we have developed a series of tools that make it simple for teams throughout Twitter to customize, develop, access and share embeddings

Example 1: User Engagement Embeddings

The graph of user content engagements holds an enormous amount of information about Twitter users. We can model this graph by separating users into "consumers" and "producers" [1], and generating embeddings for each consumer and producer such that consumer u_i's likelihood to engage with producer u_j's posts is proportional

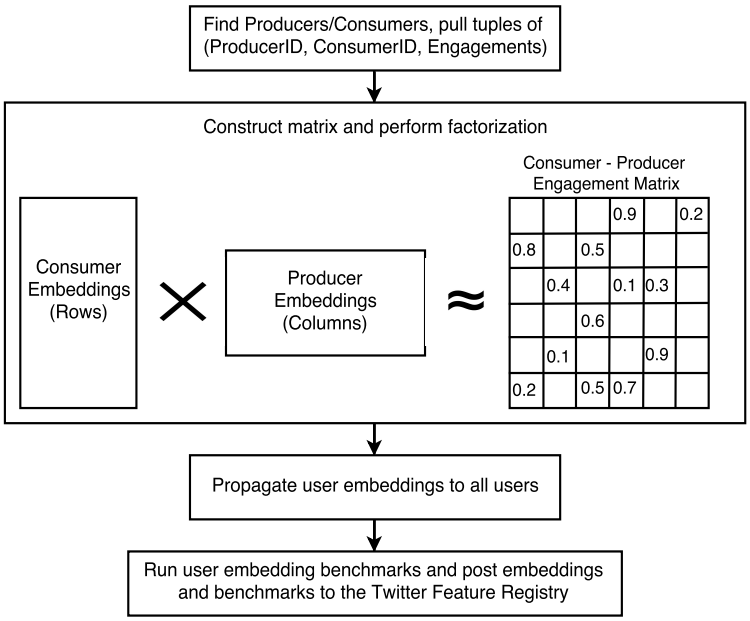

Figure 1: The ML workflows platform joins the engagement embedding steps into a configurable and reusable pipeline.

to the similarity between u_i's embedding and u_j's embedding. We generate these embeddings with the following pipeline (Fig 1):

(1) Run a data aggregation job that identifies producer users and highly active consumer users[2] and accumulates all consumer-producer engagements to form the sparse engagement graph.
(2) Represent this graph as the sparse NxM matrix A, where each row represents a consumer, each column represents a producer, and element (i, j) contains the number of times that consumer u_i engaged with producer u_j's posts. Factorize this matrix into the Nxk and kxM matrices X and Y such that $XY \approx A$ (by utilizing techniques like those in [2] or [3]). The rows of X are k-dimensional consumer embeddings and the columns of Y are k-dimensional producer embeddings.
(3) Use a folding-in technique like in [5] to propagate the consumer embeddings to all Twitter users.
(4) Run user embedding benchmarks to evaluate the quality of the embeddings and publish the embeddings and benchmarks to the Twitter Feature Registry.

These embeddings are useful for a variety of tasks within Twitter. For example, we've seen that adding engagement embeddings to email recommendation models produces significant performance

[1] The Twitter engagement graph is very lopsided, so it makes sense to model very popular users differently

[2] Filtering consumers on user activity can reduce graph sparsity and improve performance and efficiency

improvements. In addition, teams with specialized needs can modify the pipeline configuration (such as switching the engagement criteria) to generate customized embeddings.

Example 2: Skipgram Word Embeddings

The word2vec system, introduced in Mikolov 2013 [4], performs very well at generating word embeddings. However, Tweets on Twitter are written in over 80 languages, and Twitter has unique word meanings and relationships that are not found in other forms of text. In addition, there are certain types of "words" such as hashtags and user IDs that have an important role in shaping the meaning of text on Twitter, but behave differently from normal words. Furthermore, new words and terms like YOLO, ASOIF, and Covfefe are introduced to the Twitter vernacular daily, and the meanings of existing words change almost as quickly. Our word vector embedding pipeline addresses these issues in a customizable fashion by regularly executing the following pipeline:

(1) Pull recent Tweets (optionally filtered by quality, length, etc and concatenated into conversations), identify frequent words and phrases and generate cross-lingual skipgram word pairs with configurable downsampling, window size, etc.
(2) Pass these pairs to the Cortex Generic Skipgram pipeline, a customizable TensorFlow pipeline that generates embeddings for entities from (token, positive) skipgram pairs.
(3) Run word embedding benchmarks and publish the embeddings and benchmarks to the Twitter Feature Registry.

Word vector embeddings are used heavily throughout Twitter. Modeling teams like abuse generation and recommendations use them as components of Machine Learning pipelines, and they also show remarkable success in tasks like keyword expansion.

Generating and Hosting Embeddings

Embedding generation pipelines often consist of a sequence of steps that can be difficult to maintain and reuse (Fig 1). In order to address this, we implement them within the Twitter ML Workflows platform - a system built on top of Apache Airflow that links data processing and ML components into reusable pipelines that can be configured with a web interface. This makes it much easier for teams to share steps between pipelines, keep embeddings up-to-date, and modify pipelines to publish customized embeddings.

In order to share trained embeddings between teams, embedding pipelines publish freshly trained embeddings to the Twitter Feature Registry, a library for ML feature data at Twitter. This makes training and launching models that utilize embeddings require no more effort than would be required for models that use any other kind of feature (Fig 2).

Benchmarking

Unlike with a classification or regression model, it's notoriously difficult to measure the quality of an embedding. One of the reasons for this is that different teams use embeddings differently. For example, while some teams use user embeddings as model inputs, others use them in nearest neighbor systems. To mitigate this problem we have developed a variety of standard benchmarking tasks for each type of embedding. Every time an embedding is retrained it is automatically reevaluated on these benchmarks, and the results are published along with the embedding.

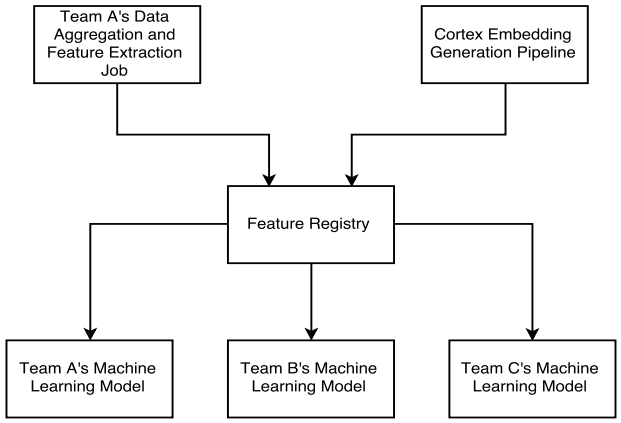

Figure 2: The Twitter Feature Registry abstracts the complexity of feature generation, so even complex features like embeddings can be easily accessed and reused.

- **Example: User Topic Prediction** During onboarding, Twitter users may indicate which topics interest them. The AUROC of a logistic regression trained on a user embedding to predict those topics is a measure of that embedding's quality.
- **Example: User Follow Jaccard** We can estimate the similarity of two users' tastes by the Jaccard index of the sets of accounts that the users follow. Over a set of user pairs, the rank order correlation between the users' embedding similarity and follow set Jaccard index is another measure of that embedding's quality.
- **Example: Word Embedding Analogies Task** For each analogy in a list of "$word_1$ is to $word_2$ as $word_3$ is to $word_4$" analogies we compute the vector $embd(word_1) - embd(word_2) + embd(word_3)$, and search for the N word embeddings closest to that vector. The percent of time that $word_4$ is among those N words is a measure of that embedding's quality.

Future Developments

We are working to build even more powerful embeddings systems in the future. For example, we are in the process of developing a generic pipeline for matching users and items with a deep coembedding network and a candidate generation system (similar in spriti to [1] and [6]). This will dramatically simplify tasks like timeline construction and ad serving, where we must generate a set of items that we believe a user would want to interact with.

Discussion

As organizations work to take advantage of the benefits of Machine Learning, it's important to decrease duplication of efforts by sharing utilities, resources, and models across teams and company verticals. Tools that facilitate this collaboration, like systems for sharing embeddings or building, tracking and launching models can abstract away the complexities of model building and allow Machine Learning to scale.

REFERENCES

[1] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. New York, NY, USA.

[2] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM '08)*. IEEE Computer Society, Washington, DC, USA, 263–272. https://doi.org/10.1109/ICDM.2008.22

[3] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*. ACM, New York, NY, USA, 426–434. https://doi.org/10.1145/1401890.1401944

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). arXiv:1301.3781 http://arxiv.org/abs/1301.3781

[5] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*. Citeseer, 27–28.

[6] Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2017. StarSpace: Embed All The Things! *CoRR* abs/1709.03856 (2017). arXiv:1709.03856 http://arxiv.org/abs/1709.03856