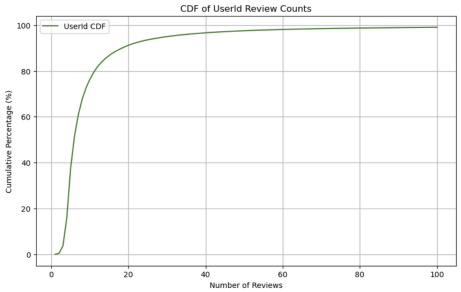
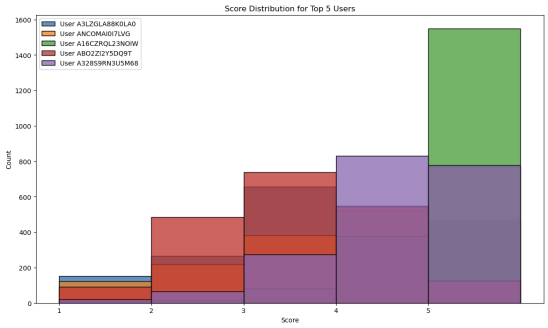
Fuzzy n-grams & Matrix Factorization

Nathan Clark October 28, 2024

Exploratory Data Analysis



I first started by analyzing the user/product ids, immediately noticing that (1) every user/product id in the test set exists in the train set, and that (2) the distribution of users/products is largely towards those with more reviews. Further, the distribution of scores for each reviewer or product varies considerably from one to another. This completely upended my initial intuition of the data, forcing me to strongly consider product/user information in my model.



Further, helpfulness also showed modest but considerable correlations with score. There were some unusual outliers in the ratio above 1 which I preprocessed to restrict to the bound of [0, 1], and the distribution suggested encoding the ratio and the log of the denominator was most useful.

While time showed correlations at large, these were weak, and there was barely any correlation in a spectral plot over weekdays, days of the month, etc.; further, I could not realize any performance benefit from including this feature.

Model Architecture

Let's begin by considering pure user/product approaches (available in archive/user_product_approach), which all take from the idea of matrix factorization. Essentially, the core concept of our technique is to assume each user and product is represented by some latent vector, and optimize latent vectors such that some operation on them most approximates the data. While I began with a custom gradient descent approach, on a review of the literature I found ALS[1,2] (alternating least squares) to be a standard efficient optimization; I leverage the implicit[3] library's implementation of ALS to accelerate this process.

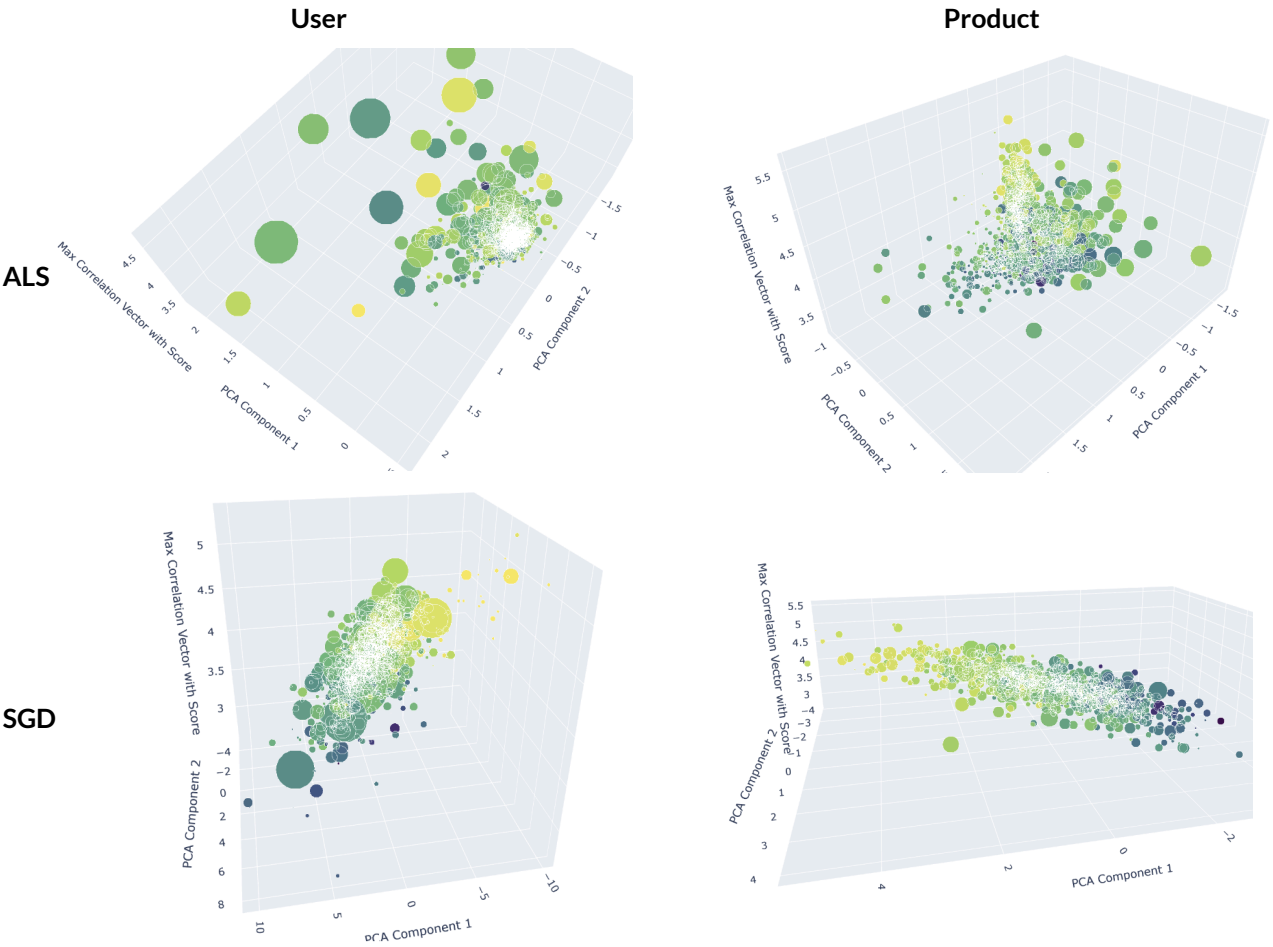
Regarding text, vector embeddings intuitively seemed like the most reasonable option; so, I chose GloVe[4] and ablated all but the 1.5 million most words, specifically the Common Crawl 840B token[5] model. GloVe is an unsupervised embedding which enables us to represent each word in our text and summaries by a 300-dimensional vector. I use an n-grams-like approach over this to extract the presence of various phrases, returning a vector indicating the presence of each n-gram.

I was surprised to see the helpfulness features I preprocessed as described above demonstrated considerable performance gains in my early tests, so I also include those directly as features.

The product/user latent spaces, text/summary n-gram presence vectors, and helpfulness features are then all concatenated; while I implemented a variety of approaches for classifying this feature vector my optimal submission simply used a dot product / cosine similarity between five trainable vectors each representing 1 through 5 stars, and chose the closest vector to as the classification decision.

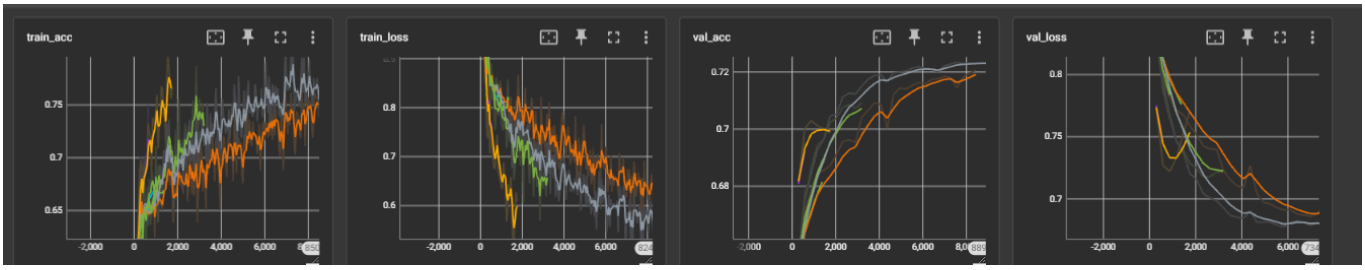
Matrix Factorization Embeddings Analysis

Interestingly, the embeddings generated by matrix factorization appeared robust to different hyperparameters and displayed distinct patterns depending on the optimization approach. In the below charts (interactable via experiments/als_exploratory_analysis.ipynb), the patterns in the embeddings between ALS (alternating least squares) and standard SGD (stochastic gradient descent) are particularly striking.



PCA of user/product embeddings with color proportional to average review, size proportional to number of reviews

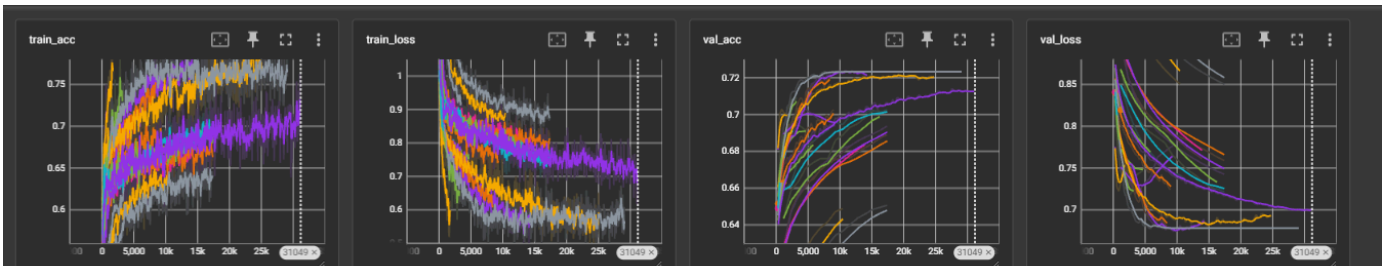
Whereas with ALS, the embedding gradually increases in norm in some distinct direction with almost no user and minimal product correlation with their means, pure SGD embeds means not only as an important metric but as one almost exactly proportional to the principal component (PCA component 1 in the above charts). Further exploration of this data suggested that the pure ALS prioritizes strictly encoding user/product relations for those with sufficient reviews whereas SGD gives a more general overview.



Experimenting with these techniques with the additional features of our n-grams model, we see that pure ALS (yellow) leads to a model that quickly achieves over 70% accuracy but dramatically overfits, whereas a pure gradient descent method (red) eventually beats ALS in validation accuracy while not learning the train data as well.

Weight Mixing

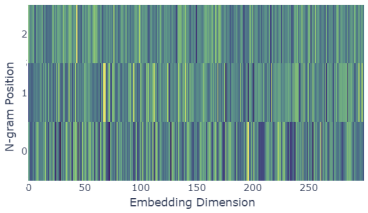
Ideally we would like a happy middle ground between the two — a fast learner which benefits from the considerable power and speed of the ALS implementation while still maximizing performance. To accomplish this, I experimented with weight mixing approaches[6] and highly nonstandard custom training schedulers, arriving at a parametrized design with a blend factor between ALS/SGD and a freeze factor which forces the model to learn single components at a time.



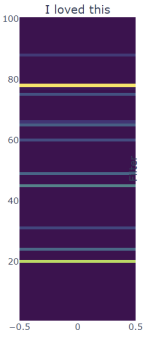
A group of experiments with different hyperparameters; the freeze factor approach is in maroon with the highest val_acc

Note the best-performing maroon model vs. the gray model's validation accuracy — the freeze factor approach learns much more slowly in a much more jagged way, but briefly barely beats the traditional model in performance. Given the unusualness of this approach, and the relatively limited time I gave towards training/experimenting with it, I was surprised to see it get the highest validation accuracy of my models.

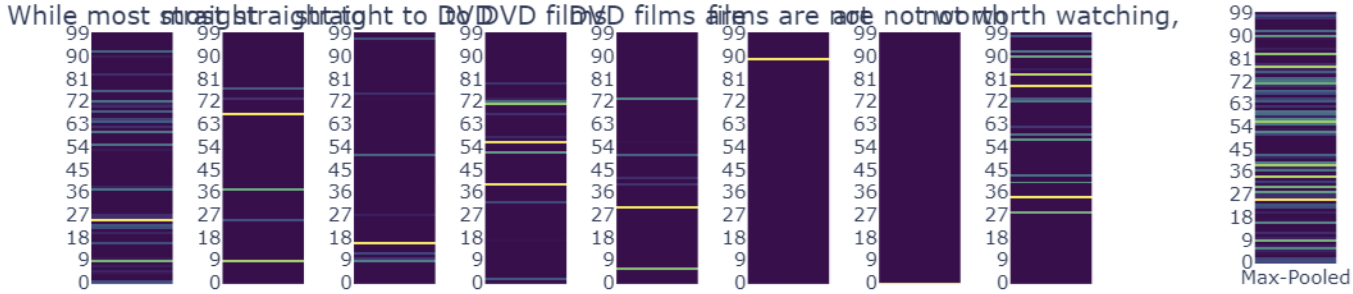
Fuzzy N-grams Analysis



Consider a selection of phrases of n contiguous words: we could use our GloVe embedding to produce an n by 300 matrix representing that n-gram. (See example 3-gram to the left.) We then can use the cosine similarity of arbitrary phrases to get a phrase similarity, or fuzzy n-gram matching. By matching to many n-grams, we can get a state space representing how closely a phrase matches to a group of n-grams, like the example shown on the right.



By matching each contiguous sequence of n words, we can match across all of our n-grams, and accumulate information via a max-pooling representing the entire text altogether.



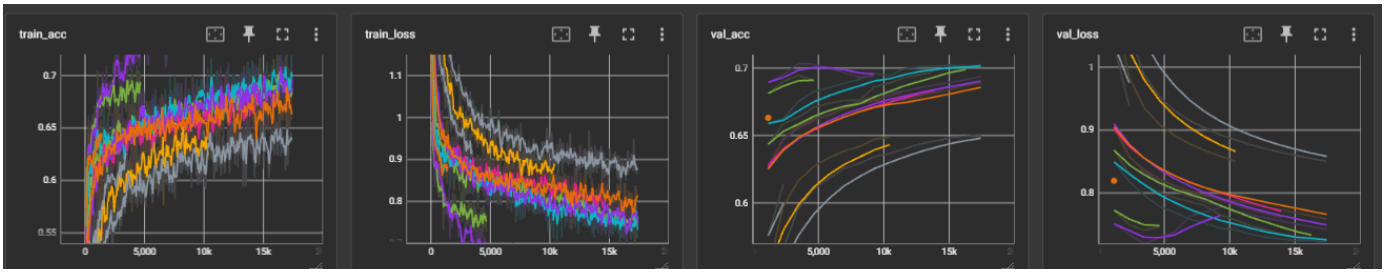
This approach gives surprisingly general yet explainable results: there exist clear correspondences between phrases and fuzzy n-grams. I realized this immediately when I tested the most salient n-gram in the model, measured by its strong contribution to the 1-star vector: it encodes a variety of obscenities that match our intuition as to the sensible contribution of obscenities to the final score, further analysis is available in experiments/n_grams_analysis.ipynb. The model does overfit to some n-grams and due to the max-pooling has limited capacity to contextualize particularly strong phrases, e.g. there exists a 2-star sample that is misclassified due to containing "3 stars." More aggressive pruning and better handling of subwords/out-of-vocab words might make finetuning the word embeddings (e.g. GloVe) more computationally feasible, enabling word representations that better match the task.

Citations

1. http://yifanhu.net/PUB/cf.pdf
2. https://www.semanticscholar.org/paper/Applications-of-the-conjugate-gradient-method-for-Tak%C3%A1cs-Pil%C3%A1szy/bfdf7af6cf7fd7bb5e6b6db5bbd91be11597eaf0?p2df
3. https://github.com/benfred/implicit
4. https://nlp.stanford.edu/pubs/glove.pdf
5. https://nlp.stanford.edu/data/wordvecs/glove.840B.300d.zip
6. https://arxiv.org/pdf/2311.07575

Appendix

Further charts are included here.



Random hyperparameter searches demonstrate the robustness of this technique over vastly different initial conditions. Unfortunately, due to time/compute limitations, more work is needed to find optimal hyperparameters.

A full example of the extraction of GloVe-embedded data is presented above. There is a 1-to-1 correspondence between the dimensionality of GloVe-embedded sets of words and the fuzzy n-grams.