Learning to be Relevant

Evolution of a Course Recommendation System

Shivani Rao, Konstantin Salomatin, Gungor Polatkan, Mahesh Joshi, Sneha Chaudhari, Vladislav Tcheprasov, Jeffrey Gee, Deepak Kumar
sirao,ksalomatin,gpolatkan,majoshi,snchaudhari,vtcheprasov,jgee,dekumar@linkedin.com
LinkedIn Corporation

ABSTRACT

We present the evolution of a large-scale content recommendation platform for LinkedIn Learning, serving 645M+ LinkedIn users across several different channels (e.g., desktop, mobile). We address challenges and complexities from both algorithms and infrastructure perspectives. We describe the progression from unsupervised models that exploit member similarity with course content, to supervised learning models leveraging member interactions with courses, and finally to hyper-personalized mixed-effects models with several million coefficients. For all the experiments, we include metric lifts achieved via online A/B tests and illustrate the trade-offs between computation and storage requirements.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Supervised learning by classification**; • **Applied computing** → *Learning management systems*; • **Social and professional topics** → Informal education;

KEYWORDS

Recommender Systems; Machine Learning; Logistic Regression; GLMix

ACM Reference Format:
Shivani Rao, Konstantin Salomatin, Gungor Polatkan, Mahesh Joshi, Sneha Chaudhari, Vladislav Tcheprasov, Jeffrey Gee, Deepak Kumar. 2019. Learning to be Relevant: Evolution of a Course Recommendation System. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM'19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3357384.3357817

1 INTRODUCTION

Online learning platforms host thousands of courses and each year new courses are added to cater to a wider audience. As the number of courses provided by a learning platform increases, there is a need to narrow the list of courses to what is deemed relevant to the *learner*[1]. Showcasing courses that are relevant to *potential* learners also helps attract new learners to the platform, thereby reaching a larger audience. Thus, course recommendation algorithms that help learners discover the relevant content are an integral part of such learning platforms.

In this paper, we present the lessons learned while building a recommender system for *LinkedIn Learning* – an online platform that aims to enable life-long learning for all LinkedIn members. The algorithms and systems that power course recommendations for LinkedIn Learning were built over a period of 3+ years and serve 645M+ members. While the theory of recommender systems is well-studied in academia, from a practitioner's perspective, there are several engineering and modeling challenges to overcome while building a recommender system in stages:

- For new users or new courses, there is limited or no labeled data for training models in the initial stages of a product – the classical *cold start* problem for recommender systems. In addition, for our system, there were limited content understanding capabilities mapping the LinkedIn Learning content catalog (e.g. courses, videos) to various entities in the LinkedIn economic graph, such as skills. In §3, we present algorithms that work under these constraints.
- After the initial launch, member interaction data for Learning content became available. It was then possible to train supervised models that can predict if a member will engage with a given course. Even with the availability of labeled data, building scalable infrastructure that supports fast iterative modeling velocity is a challenge in itself. Additionally, after the launch, we served course recommendations to various new acquisition channels to attract new learners, thereby increasing the amount of data generated by our algorithms and the amount of computational resources needed. In §4, we present our supervised learning platform and the optimization efforts for scaling this platform. This platform allowed us to test 60+ model variants, thereby leading to significant insights and improvements in metrics within a short period of time.
- Following the availability of a robust supervised training platform along with considerably large amount of engagement data, we focused on building more complex models that capture member-level interests and hyper-personalize the recommendations using mixed-effects models with several million coefficients. We present these advances and related experiments in §5.

[1]Learner is a user or a member who is in the process of learning via the online learning platform. In this paper we use *learner*, *user*, and *member* interchangeably.

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357817

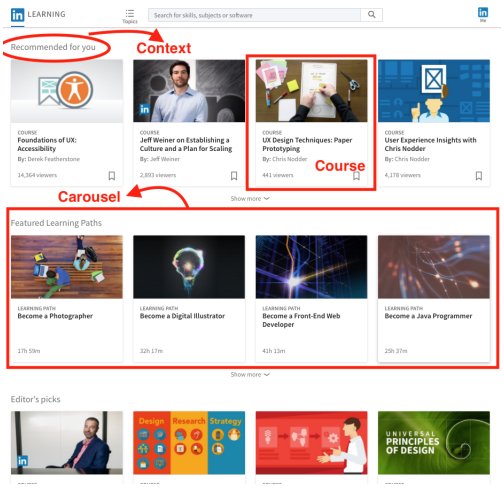

Figure 1: LinkedIn Learning Desktop landing page

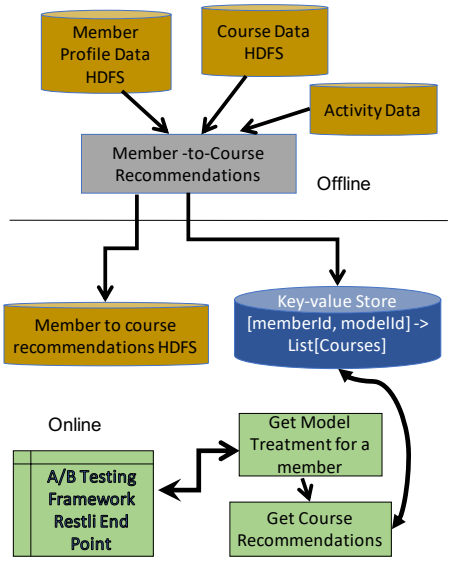

Figure 2: Overall Architecture of our Course Recommendation Engine

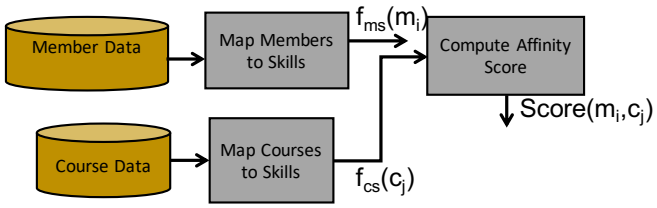

Figure 3: Skills-based Course Recommendation Model

2 SYSTEM OVERVIEW

The goal of our course recommendation system is to predict if a course c_j will be relevant to a member m_i. The overall architecture is shown in Figure 2. There are two layers to our architecture: online and offline. The actual ranked list of recommendations for each member is computed offline and stored in an online key-value store [5], queried at request time. This approach is in contrast with online scoring, where the ranked list is computed at request time. The advantages of offline scoring are its simplicity, reliability, and low latency of the serving component. However, it is challenging to scale, and suffers from an inability to use session-level contextual information. For channels such as the LinkedIn Learning landing page, this limitation is not very critical because the learning interests of members do not change significantly between sessions and our library of courses is not as large and dynamic as, for example, a collection of news articles [9].

In the online portion of our architecture, recommendations are delivered to the front-end via an online REST endpoint. The online flow ensures that the right recommendations are delivered for the right members by leveraging the A/B testing framework developed at LinkedIn. Although not shown here, the online REST endpoint is designed to provide functionality like blending in recommendations from real-time signals (such as the skills followed by learners on the LinkedIn Learning landing page at on-boarding time), filtering out dismissed courses and so on.

3 RECOMMENDING WITHOUT LABELED DATA

Course recommendation algorithms employed by most online learning platforms rely upon behavioral data for existing users in order to make high quality recommendations. When user activity (browse, click, viewing) is available, it provides implicit signals that can be used to train a supervised classification model, or a collaborative filtering model. For new users or new platforms, such signals are absent and this creates an on-boarding problem, also known as the *cold start problem* [10]. One way to mitigate this problem is to use an explicit questionnaire asking members to *follow* the skills they are interested in at the time of on-boarding. While we leverage the explicitly *followed* skills a member picks at on-boarding time to make course recommendations, we don't rely entirely on it. We leverage the plethora of information about members that is available via their LinkedIn profile and their professional network in order to generate course recommendations even before members have been on-boarded into the learning platform. Our approach is to use skills as a representation space for members as well as courses (See Figure 3). In the following sections we will describe our approaches to mapping courses as well as members to the skills space.

3.1 Tagging Courses with Skills

A course is a relatively new entity in the LinkedIn eco-system and hence, we had to build this mapping from scratch. We first leverage, the taxonomist (human taggers) generated course-to-skill[2] mapping and later use this information to build models that can automatically tag courses. In what follows, we present various

[2]We use the term "course-to-skill" to refer to the mapping that associates each course to a set of skills that the course is designed to teach.

techniques for inferring the course-to-skill mapping, and compare these techniques.

course to skill via taxonomy (cs01) . All courses in the LinkedIn Learning system are tagged with categories, and these categories map to standardized LinkedIn skills. We start with extracting a course-to-skill mapping via the help of taxonomists. Taxonomists use a two-step process to create these mappings. The first step involves, mapping all courses to skills via the category-to-skill mapping, and the second step involves manual curation of these tags. This approach results in a high precision human-generated course-to-skill mapping, but has poor coverage (only 2% of LinkedIn skills are covered by this mapping).

course to skill via skill taggers (cs02) . We also leveraged LinkedIn's automated skill taggers [1] to extract skill tags from course data. These skill taggers are trained on member profile data. While these automated skill taggers improve coverage, they are prone to a higher false positive rate than what is acceptable for our use case. Hence, we don't use this model directly, but instead train a supervised model using the output of the skill taggers as features, which is described below.

course to skill via supervised model (cs03). Given the high precision labels from the taxonomist and the skill tags associated with a course (extracted from textual information found in title, description, and video transcripts), we built a supervised binary classification model that predicts whether for a given (course, skill) pair the skill is relevant to the course. The labeled training data is derived by treating the human-tagged (course, skill) pairs as positive examples. Random negatives are added to create a balanced dataset. The features used are binary, and indicate if a match exists between the skill and the course's textual components (title, description, categories, section names, video names etc) based on the skill tags extracted by the skill tagger. For each (course, skill) pair, a pre-existing skill-to-skill similarity mapping is also used to compute additional binary features for each of the related skill (using the approach described above). The model thus learns to predict if a skill s_k is relevant to a course c_j. The drawback of this approach is that (a) it relies heavily on the quality of skill taggers (which we know is poor to start with), and (b) it trains one single logistic regression model for all the (course, skill) pairs, which is not able to effectively capture the per-skill level effects.

course to skill via semi supervised learning (cs04). This approach slightly differs from the previous approach:

- We utilize only video transcripts (obtained from the videos that belong to the course) instead of all the other short forms of textual information available from the course metadata. This is because video transcripts are long and more descriptive, which helps to capture the actual content of the course.
- We learn a separate model for each skill where the prediction is made for each course, as opposed to one common model for all (course, skill) pairs. This is for capturing the effects specific to each skill.
- We utilize various label generation techniques to augment the positive labels coming from the taxonomists. In particular, we use label propagation and label augmentation via a skill-correlation graph. This step is in addition to the use of random negatives as before. Adding positive labels in a controlled way allows for automatic discovery of new relevant skills.

The main advantage of this approach is that it does not rely on the skill taggers and it builds a per-skill model.[3]

3.1.1 Evaluation. We used two offline metrics to evaluate our course-to-skill mapping: (a) Skill Coverage (b) Precision and Recall. Skill coverage measures how many LinkedIn standardized skills are covered by the mapping. Precision and Recall are evaluated against the taxonomist-tagged course-to-skill mapping.

Skill-Coverage Table 1 shows skill-coverage for each of the course-to-skill mappings. As can be seen from Table 1, supervised models have higher coverage than the taxonomist-tagged mapping.

Precision and Recall By treating the course to skill mapping from taxonomist as ground truth, we compute precision and recall on the test set. Note that cs04 has higher Precision and Recall than cs03, but lower skill coverage. This means that the model cs04 has lower false positive rate.

Mapping Type	cs01	cs03	cs04
Skill-Coverage	2%	17%	10%

Table 1: Skill-Coverage of the various course to skill mappings: cs01 (taxonomist labeled), cs03 (supervised model) and cs04 (semi supervised model)

Table 2: Course to skill model performance

Models	Precision	Recall
cs02 (skills tagger based)	0.001	0.112
cs03 (supervised model)	0.309	0.206
cs04 (semi-supervised model)	0.327	0.231

3.2 Tagging Members with Skills

Members can be mapped to skills in various ways given the rich source of information that LinkedIn has from their profiles, such as their job title, function, industry, endorsements, network and the LinkedIn Economic graph [8]. We present three approaches for computing these mappings and measure *coverage* as one of the criteria to measure the quality of the mapping. Skill-coverage means how many of the total skills that LinkedIn has are represented in the mapping. High skill-coverage means better representation in the overall model. Similarly, member-coverage measures what percentage of members are mapped to skills space.

member to skills via profile (ms01). LinkedIn members can add skills on their profile by either entering free–form text, or choosing one of the available auto–complete options. Free form text skills entered by members need to be standardized (i.e. mapped to one of the skills available in the LinkedIn skills taxonomy). The

[3]For more details about the work, please refer to the NIPS 2017 Workshop paper presented at Learning with Limited Labeled Data [3].

standardization team at LinkedIn uses a learned model to map these free–form text skills entered by members to a standardized set of skills [1]. The learned model also provides a score for the mapping, which indicates the probability that the member has the given standardized skill. This mapping only covers 30% of the member base.

member to skills via title and industry (ms02). While the set of skills available on a member's profile is a great way to obtain a member-to-skills mapping, not all profiles have this information filled up accurately, or completely. In such cases, it becomes important to rely on other sources of such information. One of the ways to infer skills for these members is via the *cohort* they belong to. For example, one cohort can be all the members that have the same title and work in the same industry. The standardization team at LinkedIn has created a mapping from {title, industry} → skills can be utilized in cases where a member has incomplete skills section, but has a job and title specified on their profile. Let the mapping of a member m_i to skills based on their title (t) and industry (d) be denoted by $f_{td}(m_i)$. The approach we employed was to compute a weighted combination of the skill-mapping available from the member's profile (*ms01*) from their title and industry. Let δ_1 indicate the weight given to skill-mapping from *ms01*, and δ_2 be the weight given to skill-mapping from title and industry. Then the new member-to-skill mapping can be obtained by the following:

$$f_{ms02}(m_i) = \delta_1 f_{ms01}(m_i) + \delta_2 f_{td}(m_i) \quad (1)$$

When a member profile's skills section is left empty, we can use their job title and industry to derive skills. The resulting mapping *ms02* has twice the member-coverage compared to *ms01*.

3.3 Experiments with Cold-start Models

For a brand new learning platform, the primary goal of course recommendations is to acquire new learners by showcasing the relevant portion of the catalog. LinkedIn's 645M+ member base gives us a wide audience to reach out to using various channels like the feed, email, profile views and so on. Since the LinkedIn Feed is the landing page of the website – the first page that appears when a member logs in, it has become the largest playing field for model evaluation. We have used the A/B testing framework developed at LinkedIn for model comparison. The models created for experimentation are shown in Table 3.

Insight 1: Affinity Models have limitations, there are diminishing returns of improving the member-to-skill and course-to-skill mappings. Affinity models are better than non-personalized models, as can be seen from Row 1 of Table 4, however, they have their limitations. With each model iteration shown in rows 2-5 of Table 3, we saw improvements in Coverage and CTR, but the improvements diminished ultimately, which exposed the limitation of cold-start models. As shown in Row 4 of Table 4, the improvements are minimal with no statistical significance.

3.4 Context Annotation

Along with relevant course recommendations, we provide high-level *explanations* about why a recommendation was made. There are two benefits of having these explanations (we also call them *reasons* or *context*): first, they improve the acceptance of the recommender system results by users [4]; and second, these explanations can be used to organize recommendations into groups or themes to improve navigation on the landing page (see Figure 1). Courses on the landing page are organized in *carousels* where each carousel contains courses that have the same context, and is annotated with that reason.

ModelId	Model Description
m001	Recency based course recommendations (non-personalized)
m002	cs01 (course to skill via taxonomy) + ms01 (member to skill via profile)
m003	cs01 (course to skill via taxonomy) + ms02 (member to skill via title and industry)
m004	cs03 (course to skill via supervised) + ms02 (member to skill via title & industry)
m005	cs04 (course to skill via semi supervised) + ms02 (member to skill via title & industry)

Table 3: Various models that we tested in Feed

Models compared	Coverage	CTR	p-value
m002 vs m001	44.8%	+51.4%	<1e-16
m003 vs m002	+11.0%	+6.2%	0.01
m004 vs m003	+5.0%	+5.0%	<1e-16
m005 vs m004	+1.2%	+0.4%	>0.5

Table 4: A comparison of how various models (shown in Table 3) performed in the Feed.

For each course recommendation c_j for a member m_i, our context annotation algorithm selects the best reason r out of the list of possible reasons R (see Figure 4 for some examples).

The algorithm works in two stages: First, for each of the reasons $r \in R$, we compute a *reason score* for each member-course pair (denoted as $f_r(m_i, c_j)$). In the second stage, we compute the best context or reason for a given (m_i, c_j) pair. Assuming the scores of all reasons $r \in R$ are on the same scale, we simply select the reason that yields the highest score: $r^*(m_i, c_j) = \arg\max_r f_r(m_j, c_j)$ [4]. Figure 4 shows some sample context annotations. Post-launch, we added several new contexts based on the activity data. New contexts included "Trending in your title/industry/company", "Trending globally", "Popular" and so on.

Insight 2: Providing context for recommendations improves the perceived relevance of the items recommended. Providing contexts for course recommendations was a significant engineering and modeling effort. In order to quantitatively evaluate how useful this feature would be in attracting learners to the learning platform, we used the the email channel. We measured the send-to-open rate

[4] The final step is common for all the models: non-parametric scaling to make the scores comparable. We use rank-based scaling, i.e. transform raw scores to inverted ranks. For a scoring function $g(m_i, c_j)$ we rank the scores for a given user and all the courses in the collection. If $g(m_i, c_1) \geq g(m_i, c_2) \geq \ldots \geq g(m_i, c_n)$ is the rank order of courses for that context, then we replace the original score with its inverted rank $f : g(m_i, c_k) \rightarrow 1/k$. This maps an arbitrary set of scores into (0, 1] range.

Because of the skill you have For each member-course pair we find the skill that contributes the most to the affinity score and use this contribution as a *reason score*.
Short relevant This *reason score* is simply the inverted duration of the course (shorter courses score higher).
Editor's pick corresponds to a small subset of courses picked by editors, in our current algorithm this explanation overrides other explanations when available.
Recommended for you this is a special meta-reason, that combines all the reasons above.

Figure 4: Some Examples of Context Annotations

for emails with and without context annotations to test the impact. The weekly recurring email send-to-open rate improved by 50% when provided with context for recommendations.

4 SUPERVISED 1.0 - BUILDING SCALABLE LEARNING PLATFORM WITH LINEAR MODELS

Post-launch, we had behavioral data for users, for example, courses watched or bookmarked by members, and courses clicked in the feed. A wide spectrum of methods that we could not use during the cold-start phase were now available to us: supervised prediction methods, collaborative filtering, and trend detection algorithms. This required a significant change to our recommendation framework. In this section we describe how we improved our course recommendations via building a supervised platform.

Developing a scalable, maintainable, and extensible supervised learning platform for a recommendation system incorporates both engineering and data oriented challenges. There is no single best solution and the design greatly depends on the goals, the scale of the problem, the available infrastructure, engineering resources, and the learning algorithms used. The design we explain in this section provided us the following benefits:

- **Pace of iteration**. In one year we tested 60 model variants in production, 7 of those iterations were major successes, and about 15-20 had statistically significant improvements.
- **Flexibility and modularity of the system**. We have the ability to quickly add new components and run custom experiments to test new ideas. Adding new learning algorithms to our platform (Logistic Regression, GLMix, Gradient Boosted Trees, Collaborative Filtering, their combinations) is a very straightforward process. From an infrastructure point of view, feature development process is separated from modeling and model training.
- **Adaptability.** We serve course recommendations in many unique channels: desktop and mobile learning home pages, desktop and mobile feed, profile pages and email. Our platform can train and serve customized recommendations for individual channels.

Our fundamental principle for the platform design is modularity. We modularize each component of building a supervised learning algorithm such that scientists can work on individual components while not depending on the other. For example, while new features

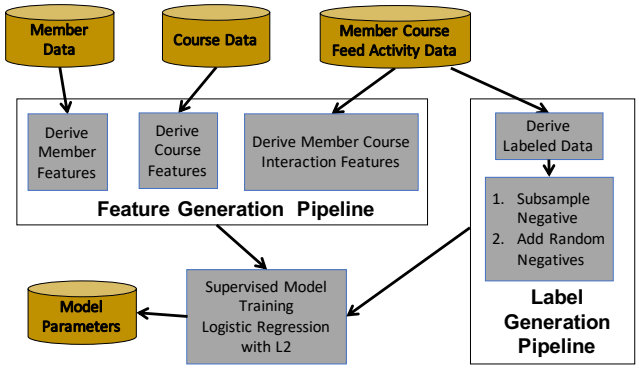

Figure 5: Supervised Training Pipeline

are being added, we can define models using different features and each might be trained with a separate learning algorithm. This was the key attribute that let us build per-channel models. Next, we will talk about each block of the platform as shown in Figure 5.

4.1 Training

Training is done offline, we use a combination of Map/Reduce and Spark jobs to prepare data and train the models (figure 5). There are three major components to training: (a) Label Generation (b) Model Optimization and (c) Feature Generation Pipeline.

Label Generation and Model Optimization. Training starts with label collection: clicks, impressions, watched courses are extracted from members' activity logs for different channels and platforms (mobile vs desktop) for a selected range of dates. The training set choice has a very strong impact on the performance of the model, so this step is very flexible. After label collection we preprocess the data: this includes splitting into train/test/validation sets, optional downsampling of negative labels, injection of random negative examples and featurization. Last step is model optimization which involves training of generalized linear models. We use an in-house open-source library Photon-ML [12]. It is built on top of Spark and allows efficient distributed training on large datasets.

Feature engineering. We created a repository of named and date-versioned member and course features, stored in hadoop file system with simple access API. These features are computed daily by hadoop jobs. Most of these jobs are simply extracting data from activity logs and database snapshots whereas some others perform complex computations, such as embeddings of course transcripts. A modeler using these features does not need to know these details. They simply provide names of the features to be used in the model with optional transformations (cross operations, cosine similarity, dot product etc). The featurization component will then produce featurized datasets. For training, the date of a data point (e.g. click) is used to extract the right date-versioned features.

4.2 Scoring

We use offline scoring, that is, we pre-compute the ranked list of recommendations for each member and store them in an online key-value store, queried at request time. This approach is in contrast

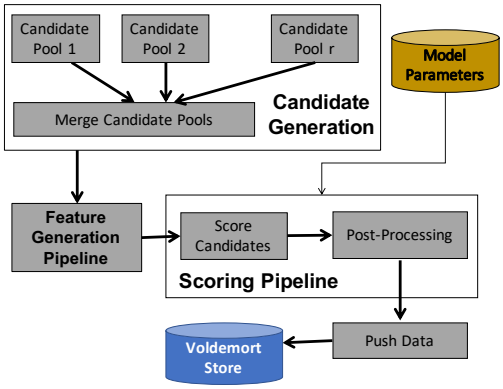

Figure 6: Candidate Generation Framework and Scoring Pipeline

to online scoring where scores are computed at request time. The advantages of offline scoring are its simplicity, reliability and low latency of the serving component. However, it is very challenging to scale (we cover techniques that we used to address this issue in section 4.3). Another limitation is its inability to use session-level contextual information. For channels such as the LinkedIn Learning homepage, this limitation is not very critical because the learning interests of members do not change much between sessions, and our library of courses is not as large and dynamic as, for example, a collection of news articles.

The scoring pipeline is outlined in Figure 6. There are three major components to the scoring pipeline: (a) Candidate Generation (b) Scoring on-the-fly using a data streaming pipeline (c) Pushing the recommendations to an online key-value store.

Candidate generation. Our algorithm selects a reasonable number (500-1000) of recommendation candidates for scoring. Originally, we adopted our cold-start skill affinity model for candidate selection. In particular, we merged the output of our high-precision skill affinity model (based on taxonomy-driven course-to-skill mapping) with a downsampled high-recall model based on skills learned by a supervised model. Later we added additional candidate pools that consisted of courses similar to the courses that the member previously engaged with.

Data streaming pipeline. This is a Map/Reduce flow that merges scoring candidates, member data, and course data, and executes the scoring function on it. This is by far the most computationally expensive piece of the pipeline. Efficient implementation of this flow is essential for scalable offline scoring. At a high level, we store course data in memory and stream member data through, performing featurization and scoring on-the-fly. Optionally, we shard the course data to fit it in memory, currently a single shard is sufficient. We replicate shards across multiple reducers, so that each reducer keeps just one shard, but each shard is replicated across many reducers. Finally, we group all the data that is keyed by member: features and list of candidates to score and send it to a random reducer that holds the course data shard.

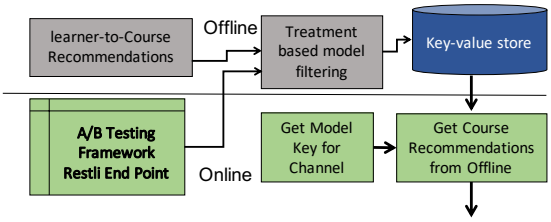

Figure 7: Offline Pre-filtering of recommendations based on the model treatments derived from A/B test rest end point

4.3 Scaling challenges in compute and storage

The overall recommendation architecture is shown in Figure 2. Each model has recommendations for 645M+ members and 200+ courses per member. However, the number of channels where the recommendations were served increased (3-5) and the number of models we experimented with in each of these channels increased (4-5) rapidly. Hence, we hit performance bottlenecks in the offline processes that push data into our online key-value store, and also reached limits of the storage capacity of the key-value store. The following are two solutions we implemented to solve the scaling challenges:

Offline Pre-filtering Infrastructure. In our original architecture (shown in Figure 2), all the recommendations are pushed to the key-value store, and the online REST endpoint returns only recommendations relevant to members based on their model treatments. However, not all recommendations are necessary. For example, if model 1 treatment is assigned to only 20% of members in channel 1, then there is no need to push the data for the remainder 80% of the members. That means, recommendations for model 1 can be pre-filtered to only the members that are assigned model 1 as the treatment. This technique results in reduction in data size and speeds up the job that pushes the data to the key-value store, and is shown in Figure 7.

Compact key-value format for efficient storing and retrieving recommendations. Another avenue of optimization opened when we paid attention to the courses that are recommended to the members by these various models : the set of courses recommended by various models remains more or less the same, with different rank ordering. We exploit this property, to propose a compact representation as shown in Figure 8. Our original key schema for our online store included a (memberId, modelId) pair, with the value being a list of course recommendations. In the new data format, we combined recommendations produced by different models for a given *memberId* into one record, and the value contains a union of courses. For each of the models, a list of indices that indexes into the list of courses is also returned. This approach reduces our data storage requirements by 3X and reduces the time to push our recommendations to our online store by 50%. This was a critical milestone letting us to build more customized models per-channel with reasonable cost of serving.

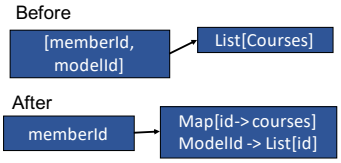

Figure 8: Compact Representation

Model #	Model Description
m101	First version of Supervised model with member features (demographics, interest, title etc.), course features (author, difficulty level, category of courses etc.) and interaction features (skill overlap, prior interaction counts on various channels)
m102	Add purchase and subscription type features (are they premium members, are they job seekers etc)
m103	Add Skills derived from jobs viewed (based on the jobs viewed by a member in the past)
m104	Add Cross Feature member seniority × course difficulty
m105	Add Cross Feature subscription type × course author and purchase/subscription features × course category

Table 5: Table summarizing various feature engineering iterations we underwent for logistic regression model

Model	Result
Our first LR model	+11% CTR
Our best LR model	+75%CTR

Table 6: Evaluation of our Logistic Regression models, compared to our best-performing affinity model m005.

4.4 Fast Experimentation and Lessons Learned

Post-launch, the goal of course recommendations on a learning platform like LinkedIn Learning is two-fold: Acquisition, and Engagement. In what follows, we present insights gained from testing 60 model variants in production using our A/B testing platform. There are mainly two metrics we measure in our experiments (a) offline ROC-AUC and (b) CTR (Click Through Rate) (since most of our testing happens in Acquisition channels where the goal is to draw learners to the platform)

Insight 3: Learned models outperform the best affinity models and continue to stay relevant. Recall from Table 4 that our affinity models plateaued and improving member-to-skill and course-to-skill mappings provided diminishing returns. This was because the model was incapable of keeping up with changing interest of the users (due to lack of features based on activity data). With learned models, we started to see improvements compared to affinity-based model from the very first iteration (see row 1 of Table 6). Since that first model up until the models used today in production we have seen major improvements in model quality and its ability to stay relevant to the learner.

Insight 4: Right training data is the most important factor in the performance of a LR model. There are numerous factors that impact the performance of the recommendation algorithm. Some of them are related to modeling, e.g. hyper-parameter values and features. Most of them are external to the model: user interface, design of the course recommendation block (layout, buttons, embedded player etc), second pass ranking algorithm that blends our recommendations with other recommendations (news, job postings, connections), seasonality, and many others. The external factors usually have a very strong impact on model behavior and metrics, which is why it is very important to collect the most relevant training data, and regularly update/retrain the model. On some occasions, training the same model on new data improved the CTR by +20%. This is documented as the "Changing Anything Changes Everything" principle by other recommender systems practitioners [7].

Insight 5: Offline metrics improvements don't necessarily mean online metrics improvements. We train our models on impressions served by previous versions of our models. This creates a dangerous feedback loop and bias in the training data. The implication is that offline evaluation (and the model itself) is subject to this bias. Several measures can be used to alleviate this issue: injection of random recommendations into sessions (training on completely randomized data as an extreme) and injection of random negative examples into the training set.

Insight 6: LR models trained on channel-specific data perform the best in this channel. Different members visit different channels with different intentions, so it is optimal to match training data distribution with serving data distribution as close as possible, *as long as you have enough training data.* However, data-hungry methods (as we mention in section 5) may benefit from all available data from all the channels.

Insight 7: Adding new features or hand-crafting cross features gives diminishing returns in metric improvements. With tried several feature engineering iterations as tabulated in Table 5 and we found that with some iterations we saw major improvements in AUC, but not so much in CTR. We attribute this to the fact that training data changes over time, and with minor changes in training parameters, we don't have an apples-to-apples comparison. In addition, hand-crafting cross-features provided diminishing returns as well in terms of model accuracy and online metrics improvements. For example, we did extensive analysis of a member's seniority level and the difficulty level of the course to understand if there was interaction between these two variables. While our offline analysis revealed that senior leaders (managers and above) tend to engage with more basic courses, and junior level individual contributors tend to engage with more intermediate and advanced level content, adding these cross features into the model gave minimal improvements in offline AUC or online CTR metrics. Same can be said for model *m105*, which added cross features between subscription type and course authors, course category.

5 HYPER PERSONALIZED MODELS

A major weakness of LR-based recommendation model from the previous section is the lack of real personalization in recommendation results. LR was able to learn useful signals from very sparse

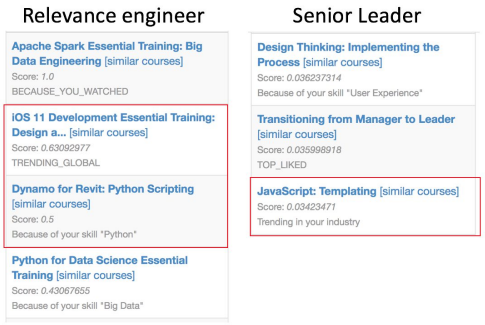

Figure 9: Table demonstrates limitations of the LR model. Bad recommendations (highlighted) of two engaged learners were ranked high because of course popularity and a skill match on old skills that members are no longer interested in.

Member × *Course* data, but does not have enough representation power to learn about individual members. As we collected more data, these limitations became more obvious, for example, our model relied a lot on quality assignments of profile skills and course popularity, see Figure 9 with examples.

To address this issue we moved to generalized linear mixed effect models (GLMix, [13]). GLMix is a model that refines Logistic Regression by extracting conditional per-member and per-course patterns. For an engaged member we learn their personal interests on their past engagements: the more the user engages, the higher the contribution of the personal model. And for courses we learn per-course models on all engagements with this course. The predicted click probability in GLMix is expressed via a sum of three components: global model, per-member model, and per-course model:

$$g\left(E\left(click|m,c\right)\right) = w^{\intercal}x_{mc} + \alpha_m^{\intercal}x_c + \beta_c^{\intercal}x_m \quad (2)$$

where g is a logit link function and the first component $w^{\intercal}x_{mc}$ represents an LR baseline. x_{mc} is a feature vector of course, member, and interaction features. The second component $\alpha_m^{\intercal}x_c$ is a per-member model. It can, for example, boost the course categories that this member engaged with in the past. x_c represents the subset of features that are useful in a per-member model, most importantly, but not limited to, course features (hence the subscript c). The last component $\beta_c^{\intercal}x_m$ is a per-course model that can boost, for example, the titles of members that previously engaged with a course.

Using a GLMix model poses a scalability challenge. The model uses billions of coefficients and requires much more data to train than LR. It has more hyperparameters to tune. Scoring is hard to scale because the model does not fit in memory and needs to be stored in a distributed way. We use an efficient open-source implementation [12] for training. Our scoring pipeline could accommodate GLMix with minimal changes, see Figure 10

The results are presented in table 7. We use ROC-AUC as our offline metric and we measure CTR gains in A/B experiments. To compare the contribution from per-course and per-member models we trained GLMix with only per-course and per-member components, we did not evaluate these variants in production. As we

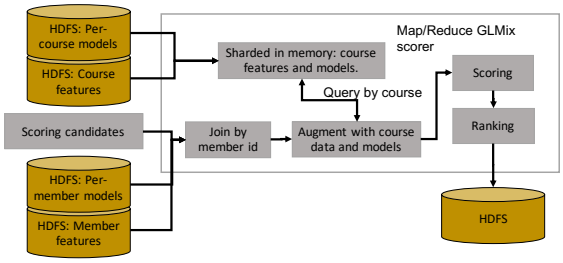

Figure 10: GLMix scoring pipeline. If we exclude per-member models and per-course models inputs, specific to GLMix, the diagram would represent our original scoring pipeline used for LR models.

Model	ROC AUC	CTR Gain
LR baseline	0.749	0
GLMix, only per-course models	0.775	-
GLMix, only per-member models	0.796	-
GLMix, per-member and per-course	**0.806**	+19.2%

Table 7: Offline and online evaluation of GLMix models.

expected, per-member models had the strongest impact with per-course models being quite useful too, especially for members with no history of engagements.

Insight 8: All member engagements matter for per-member GLMix models. Per-member models are trained only on a given member's engagement. This data is very sparse, so it is beneficial to collect as much data as possible. We used a much longer time window for data collection than for LR, and combined data from different channels. Both of these measures helped to improve GLMix performance. However, they degraded the performance of LR. We believe this is because LR does not need much data to train, and after the performance saturates on recent data from the channel, adding training data with a different distribution (e.g. older data) hurts the model.

6 ADDITIONAL EXPERIMENTS

6.1 Autoplay Engagement

Video Autoplay feature was recently enabled for the introductory video for any course recommendation in the Flagship Feed. Any course recommendation update seen by a member in their feed automatically starts playing the introductory video of the course, with volume muted, but video transcripts (sub-titles) on. Via autoplay, members can engage with course recommendations without having to perform any explicit action - no click is needed. We hypothesize that autoplay captures implicit member interest in Learning Course Recommendations based on the duration of autoplay, without the need for any explicit interaction.

There are two possible approaches to incorporating autoplay engagement: 1. encode it as additional features, or 2. encode it in the form of updated labels, by labeling impressions as positive instances based purely on autoplay engagement. We chose to model

the second approach as it is a much stronger form of input to the model.

For converting implicit autoplay engagement into positive labels, we considered several approaches:

- Fixed time threshold on autoplay duration, with various threshold values (9s, 19s, 29s, etc.)
- Threshold on the fraction of total video duration watched (0.5, 0.9)
- A dynamic, percentile threshold on aggregated viewing statistics over the last 30 days. Aggregation was done in 3 different ways - global, per-video, per-member.

Insight 9: Autoplay engagement modeled using a dynamic and personalized threshold on the duration of autoplay can provide a valuable relevance signal. With a model that uses a 95th percentile threshold on member-level watch times to consider impressions as positive instances +1.29% gain in engaged learners compared to baseline. Note that this is despite the fact that the autoplay engagement based model actually showed a lower offline AUC on the test set from the baseline model (which did not contain positive instances based on autoplay).

6.2 New Courses

A common area of concern for recommender systems is how they treat new items. Since one of the prominent features in our model is the click through rate for a course, there is a risk that the recommendation system would favor old items with more engagement.

Insight 10: Showing new courses does not always boost online metrics. We ran an experiment where we boosted the ranks of courses released in the last 30 days to increase their chance of being seen. After the boost, the average number of new courses in top 30 positions doubled. We ran an A/B test against the original ranking and did not see any statistically significant metrics change.

6.3 Contextual Non-Static Features

Originally our course recommendations were stateless, or static. We hypothesize that a learner's engagement patterns shift according to recurrent events such as commute vs. prime time in the evening, or weekdays vs. weekends.

Insight 11: Contextual and real-time features provide freshness in recommendations. We included a feature called day-of-the-week such that the recommendations for each day are different for members if such day-to-day patterns exist. Such models provide a fresh organic experience. For example, we found that the order of the recommendations changes six positions on average from day to day. Online experiments showed significant wins: +0.78% increase in engagement among subscribers, +1.08% increase in engagement among non-subscribers, +3.24% increase in CTR in feed.

7 CONCLUSION

In this paper, we present the evolution of a course recommender system for LinkedIn Learning – an online learning platform that enables professionals and students on LinkedIn to find economic opportunity via lifelong learning. We track the evolution of the recommender system as the product evolved from pre-launch to post-launch. Pre-launch, we were faced with the *cold-start* problem as we had no activity data about members on the learning site. We proposed a course recommendation approach based on skills where members and courses are mapped to the skills space. Post-launch, we were able to mine activity data and developed a scalable supervised learning platform addressing multi-channel models and supporting 645M+ members. More data also enabled us to build more complex models with hyper-personalization through algorithms such as GLMix. We have conducted experiments using our in-house A/B testing framework on various LinkedIn channels like the Feed, the weekly recurring email, and the LinkedIn Learning page and drawn insights that would be useful to other practitioners in recommender systems.

There are several avenues for future work that we are actively investing in. We are looking into including embeddings learned via deep neural networks as features into our models [2], developing graph-based approaches to bootstrap recommendations for new courses or new users [7], developing an online scoring flow that would rank recommendations in real-time utilizing information within the current session, for instance, articles read or search intent [11], and a near-line impression discounting platform [6].

REFERENCES

[1] Mathieu Bastian, Matthew Hayes, William Vaughan, Sam Shah, Peter Skomoroch, Hyungjin Kim, Sal Uryasev, and Christopher Lloyd. 2014. LinkedIn Skills: Large-scale Topic Extraction and Inference. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 1–8.

[2] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[3] Zhe Cui and Shivani Rao. 2017. Video to Skill Tagging using Transcripts under Weak Supervision. In *Workshop on Learning with Limited Labeled Data, Neural Information Processing System*.

[4] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining Collaborative Fltering Recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, 241–250.

[5] Jay Kreps. 2009. Project Voldemort. LinkedIn Blog. https://blog.linkedin.com/2009/04/01/project-voldemort-part-ii-how-it-works.

[6] Pei Lee, Laks VS Lakshmanan, Mitul Tiwari, and Sam Shah. 2014. Modeling Impression Discounting in Large-scale Recommender Systems. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1837–1846.

[7] David C Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. 2017. Related pins at pinterest: The evolution of a real-world recommender system. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 583–592.

[8] Michael Mandel. 2014. Connections as a Tool for Growth: Evidence from the LinkedIn Economic Graph. *November. http://www. slideshare. net/linkedin/mandel-linked-in-connections-reportnov-2014* (2014).

[9] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. 2017. Embedding-based news recommendation for millions of users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1933–1942.

[10] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and Metrics for Cold-start Recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 253–260.

[11] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D Convolutional Networks for Session-based Recommendation with Content Features. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 138–146.

[12] Xianxing Zhang, Deepak Agarwal, Bee-Chung Chen, and Paul Ogilvie. 2016. Building Recommender Systems using Photon ML. https://github.com/linkedin/photon-ml.

[13] XianXing Zhang, Yitong Zhou, Yiming Ma, Bee-Chung Chen, Liang Zhang, and Deepak Agarwal. 2016. Glmix: Generalized linear mixed models for large-scale response prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 363–372.