TALENT ANALYTIC STUDIO: RECOMMENDATION ENGINE FOR EMPLOYEE TRAINING COURSES AT ORANGE - SILICON VALLEY

JUSTIN J. WANG*, WILSON LAU

April 26, 2017

Abstract. Orange has the ambition to create a new learning experience through a new Learning Management System. The goal of Orange's employee training recommendation system is to advance the skillset of the company's workforce by promoting active participation and progress in completing SkillSoft courses. The Talent Analytic Studio project leverages machine learning and analytics to solve HR challenges–addressing skill gaps and empowering people to take on new projects.

Key words. Cross-Occurrence algorithm, doc2vec, precision@k, PredictionIO, Spark MLlib, hybrid recommender system, collaborative filtering, content-based recommendation engine

1. Introduction. The point of implementing this internal recommender system for training courses is to get employees to actually take recommended course, in a way that betters the employee experience through personalization. It's also to address skill mismatch for govermenent or unionized employees to update their skillsets and move to new positions.

2. Datasets. The datasets used by the recommendation engine comes from 3 sources: the SkillSoft catalog, which summarizes the offered courses; employee data, which comes from the profile and interactions within the Plazza social network; and the training record of past courses completed by employees. Orange has over 155,000 employees as of December 31st, 2016 [5]. The pilot dataset consists of 500 users.

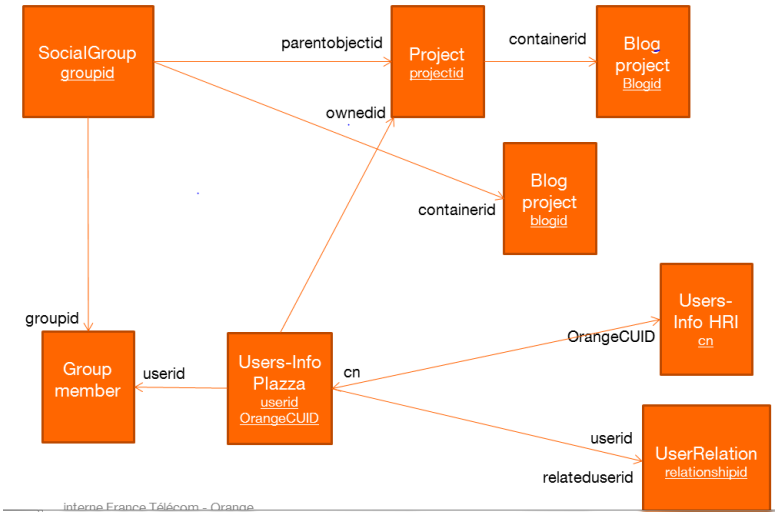

FIG. 2.1. *Schema for User-related Tables*

2.1. Database Architecture. lorem ipsum

2.2. Deploying the Engine in Production. We use internal servers to handle all the data storage and computing on premises. The data center is on the top floor of the San Francisco office. We use Apache PredictionIO (incubating) as our full machine learning stack to deploy the recommendation engine as a web service in production. The infrastructure is open source, fast and scalable. The technologies used include Spark, MLlib, HBase, Spray, Elasticsearch, Apache Mahout, Maven, Java, Scala, and Python.

3. The Correlated Cross-Occurrence Algorithm [2].

[1]primary author: Justin J. Wang, linkedin.com/in/justw, github.com/justwjr, phone: (626) 298-4825, email: justwjr@ucla.edu
[2]Orange - Silicon Valley, 60 Spear St, San Francisco, CA 94105
[3]University of New Haven(San Francisco), 44 Tehama St., San Francisco, CA 94105

Algorithm 1 Correlated Cross-Occurrence [6]

Require: spark-itemsimilarity to use multiple user actions

return current-flow betweenness approximation $c'_{CB} : V \to \mathbb{R}_{\geq 0}$

Mahout will use cross-action cooccurrence analysis to limit the views to ones that do predict purchases. We do this by treating the primary action (purchase) as data for the indicator matrix and use the secondary action (view) to calculate the cross-cooccurrence indicator matrix.

spark-itemsimilarity can read separate actions from separate files or from a mixed action log by filtering certain lines.

3.1. The Universal Recommender. The correlated cross-occurrence algorithm is a hybrid between collaborative filtering and a content-based recommender, supporting a wide variety of user input data to indicate tastes. Apache Spark MLlib's Collaborative Filtering algorithm uses the alternating least squares algorithm to learn a small set of latent factors from a single type of user action represented in a weighted user-item association matrix. Hence, pure collaborative filtering could construct a matrix based off courses taken, but not different types of events like joining a social group. In contrast, the universal recommender's cross-occurrence algorithm can accept multiple different types of user events or contextual information and is arguably the only publicly documented recommender to do this. It also supports item properties for filtering and boosting recommendations. [2]

4. Evaluation Metrics. In this section, we discuss offline evaluation techniques for the recommender system. Since the project is not yet in production, it is not possible to conduct experiments and run A/B testing, which would be the ideal way to evaluate algorithm variants to see how well changes improve meaningful metrics such as rate of course completion among a group of employees. Instead, we turn to offline evaluation techniques, some of which are root-mean square error (RMSE), mean absolute error (MAE), mean average precision at k (MAP@k), and precision at k (precision@k), which measures the portion of relevant items amongst the first k items [7]. We use a rating threshold to define whether a recommendation is relevant. 4.1

These offline, cross-validation testing methods are severly limited and some results are known to be contrary to real-world results discovered through A/B tests. Increasing data used in training will almost always increase cross-validation test results, but randomizations and tuning for recency will almost always decrease scores [1].

Offline evaluation metrics, such as precision@k allows us to tune the engine to find optimal parameters. The script used to evaluate the engine is in the file, "Evaluation.scala". For the training course recommendations, there are no user ratings for courses. Instead, the events are whether a person took a course, and whether or not the course was completed. Hence, our event types are either 1)"started and finished course" or 2)"started course but did not finish".

4.1. Doc2Vec Experiment. One of the challenges with working with this dataset is that the Skillsoft course catalog updates quarterly, where an updated course adopts a new course id. The same course ends up having multiple different course ids, which messes up the recommender system, so we needed a way to identify which courses should be considered as the same course id. A rudimentary approach is to fuzzy match on the course titles. We used the python package called FuzzyWuzzy [3], which matches strings based on the Levenshtein distance to calculate the differences between sequences. Partial results are shown in Figure 4.2.

The code and full results are contained in the notebook "Fuzzy matching course title.ipynb". Obviously, when the fuzzy score between two titles is 100%, we can confidently determine that they're the same course. For results that were not a perfect match, it may be tempting to choose a threshold score and say that match scores above 80% lets us group the two courses together; however, this rule would result in erroneous determinations. For example, we observed in the catalog, one course named "Black-Box Software Testing Techniques", and another course named "White-Box Software Testing Techniques", completely different courses whose titles differ by one word, resulting in a fuzzy match score of 86%. Matching course titles based off anything other than a perfect match becomes problematic. For instance, the prime offender to this approach are the two courses: "TestPrep 1Z0-808 Java SE 8 Programmer I", and "TestPrep 1Z0-809 Java SE 8 Programmer II", whose fuzzy match score is 97%. Title alone would not give us an automated way to find updates unless there is an exact match.

A semantic approach using Gensim's doc2vec [4] on course descriptions is an interesting way to try to identify similar courses. We scraped course descriptions and target audience information through Skillsoft URLs; the results are in the notebook, "TODO: name of notebook". Doc2vec produced really impressive results sometimes, but overall, the results were all over the place. A possible feature implementation might be

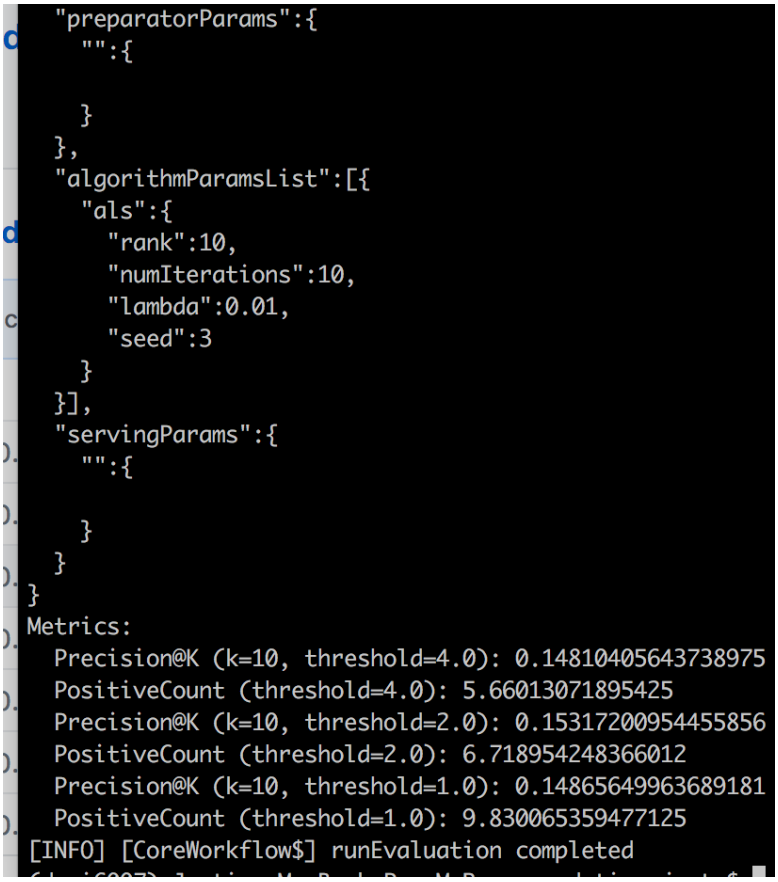

FIG. 4.1. *precision@k Evaluation Metric on the Recommendation Engine*

	StepIndex	STEP Course#	STEP Title	CatalogIndex	Catalog Course#	Catalog Title	Fuzzy Score
3897	3.0	mntpmp5ed	Mentoring Project Management Professional (PMP) PMBOK Guide 5th Edition Aligned	828.0	mntpmp5ed	Mentoring Project Management Professional (PMP) PMBOK Guide 5th Edition Aligned	100.0
4793	4.0	pd_23_a02_bs_enus	Public Speaking Strategies: Confident Public Speaking	701.0	pd_23_a02_bs_enus	Public Speaking Strategies: Confident Public Speaking	100.0
18350	18.0	cust_02_a02_bs_enus	Effective Communication Skills	959.0	cust_07_a02_bs_enus	Communication Skills	80.0
9967	9.0	proj_04_a05_bs_enus	Controlling, Managing and Closing a PRINCE2-aligned Project	760.0	ib_prin_a05_it_enus	Controlling, Managing and Closing a Project (PRINCE2?: 2009-aligned)	80.0
18244	18.0	cust_02_a02_bs_enus	Effective Communication Skills	853.0	team_02_a04_bs_enus	Effective Team Communication	79.0

FIG. 4.2. *Levenshtein Distance Between Course Titles*

to supplement the Cross-Occurrence algorithm by also recommending the course that had the greatest doc2vec cosine similarity to the top scoring course.

Here is an example of an impressive match result from doc2vec. These two course descriptions were convereted to vectors using doc2vec, and they had a cosine similarity of 0.999690:

```
Exhibit A:  Documentation and Criteria Used for Business Analysis
   ["Business analysts must develop a repository of common language to facilitate
   communication and strategically align activities and goals. In this course, you'll
   learn about a number of business analysis techniques included in the categories
   of documentation, business and user cases, and setting metrics and criteria ..."]

Exhibit B:  Business Analysis and Solution Evaluation
   ["After a solution has been partially or wholly implemented, a business analyst
   measures its effectiveness and ability to deliver the expected value to stakeholders.
```

```
        This involves measuring performance and identifying limitations or constraints that
        are keeping the solution from reaching its full value potential. The business analyst
        then recommends actions for overcoming any limitations..."]

```

FIG. 4.3. *Highest Scoring Cosine Similarity Pairing for these Course Descriptions*

In testing doc2vec, we observed, unsurprisingly, that a duplicated course description paired most similarly with itself. While doc2vec did produce some promising results, such as the one shown above in figure 4.3, there were also a high number of disappointing, nonsensical results, which occur often enough that we decided to defer implementing doc2vec into the current product. Initial results were discouraging; however, maybe with further refinement, such as a method to filter out nonsensical results, we could use semantic similarity as an add-on to the regular set of recommendations. For our purposes and the given catalog, we decided to base course-update logic on having an exact match on course title. In some cases, we could also match courses that map to the same URL, a feature that we are fortunate enough to have in our dataset.

5. Summary and conclusion. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Felis eget velit aliquet sagittis id. Pharetra sit amet aliquam id diam maecenas. Risus quis varius quam quisque id diam vel quam. Rhoncus mattis rhoncus urna neque viverra justo nec. Sem et tortor consequat id porta nibh. In nisl nisi scelerisque eu ultrices vitae auctor eu augue. Ac placerat vestibulum lectus mauris ultrices eros in. Sit amet consectetur adipiscing elit pellentesque. Tristique risus nec feugiat in fermentum. Sed ullamcorper morbi tincidunt ornare. Ac odio tempor orci dapibus ultrices in iaculis. Enim ut tellus elementum sagittis vitae et leo duis ut. Et molestie ac feugiat sed lectus vestibulum mattis ullamcorper velit. Ac turpis egestas sed tempus urna et. Mi tempus imperdiet nulla malesuada pellentesque elit eget. Malesuada pellentesque elit eget gravida cum sociis natoque penatibus et.

Non curabitur gravida arcu ac tortor dignissim convallis. Commodo nulla facilisi nullam vehicula ipsum. Purus non enim praesent elementum facilisis leo. Dui nunc mattis enim ut tellus. Consectetur adipiscing elit duis tristique sollicitudin nibh sit amet commodo. Dui accumsan sit amet nulla facilisi morbi. Iaculis eu non diam phasellus vestibulum. Ultricies mi quis hendrerit dolor magna eget est lorem ipsum. Non arcu risus quis varius. Mattis vulputate enim nulla aliquet porttitor lacus luctus accumsan. Volutpat odio facilisis mauris sit amet massa vitae tortor. Et egestas quis ipsum suspendisse. Leo in vitae turpis massa sed elementum tempus. Vitae aliquet nec ullamcorper sit amet risus nullam.

REFERENCES

[1] ActionML, *Advanced tuning: Finding the strongest user indicators, map@k*. http://actionml.com/docs/ur_advanced_tuning, 2016.

[2] ———, *The universal recommender: The correlated cross-occurrence algorithm (cco)*. http://actionml.com/docs/ur, 2016.

[3] Adam Cohen, *Fuzzy string matching in python*. https://github.com/seatgeek/fuzzywuzzy, 2011.

[4] Radim Řehůřek and Petr Sojka, *Software Framework for Topic Modelling with Large Corpora*, in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta, May 2010, pp. 45–50. http://is.muni.cz/publication/884893/en.

[5] Orange S.A., *Orange - get to know us better*. https://orange.jobs/site/get-to-know-us-better/index.htm, dec 2016.

[6] The Apache Software Foundation: Licensed under the Apache License Version 2.0, *Intro to cooccurrence recommenders with spark*. https://mahout.apache.org/users/algorithms/intro-cooccurrence-spark.html, 2016.

[7] Justin Yip, *Evaluation explained (recommendation) - predictionio*. http://predictionio.incubator.apache.org/templates/recommendation/evaluation/, 2016.