November 2020

FASTTRACK RECOMMENDER SYSTEM

Adam Jannetta

John Garrett

Ammar Rayes

Altaf Karim

FASTTRACK RECOMMENDER SYSTEM

AUTHORS:
Adam Jannetta
John Garrett
Ammar Rayes
Altaf Karim

ABSTRACT

Network operators are overloaded with numerous recommendations coming from vendors, some of which come from automated recommender systems. Such automated recommendations may or may not apply to a customer's specific environment, often lack an assessment of priority within the context of the other recommendations, and may or may not apply to an individual customer's scenario. To address these challenges, techniques are presented herein that provide a novel approach to generating and ranking recommendations coming from a dynamic recommender system where rankings are based on enriched context from, for example, live data on real networks, activities performed by real customers, etc. Such techniques enhance the operational features of existing networks by recommending popular items to new customers, identifying critical items that can be proactively addressed in order to provide additional services, and reducing Technical Assistance Center (TAC) cases when patches exist for common issues.

DETAILED DESCRIPTION

Network operators are overloaded with numerous recommendations coming from vendors, some of which come from automated recommender systems. Such automated recommendations may or may not apply to a customer's specific environment. Further, such recommendations often lack an assessment of priority within the context of the other recommendations that such a system provides as well as adoption trends of industry peers. Finally, such recommendations may or may not apply to an individual customer's scenario. Even high priority recommendations are not of value to a customer who is not running a configuration that would benefit from the recommendations.

Recommendations in this context may be sourced from, for example, vendors, experts, expert systems, or machine learning pipelines. Ultimately the customer is presented with an unwieldy list of items that can cause them to miss the important recommendations or tune out the source of the items altogether. The general professional services industry presents a large volume of recommendations to customers, and while valuable, customer feedback suggests that it is difficult to parse and prioritize the insights and recommendations.

To address these types of challenges techniques are presented herein that support a recommender system that is designed to cover any cases that meet the following criteria:

- New items become available all of the time.
- A large population is available to observe the adoption of those items.
- When the items become part of a larger set, they are no longer needed if the set itself is used.

This is a common scenario across many areas regarding customer recommendations. There is a need to add priority and ranking to the automated recommendations such that the network operator can make a decision whether to implement the recommendation in the short term, the longer term, or not at all.

For purposes of exposition, in the narrative that follows aspects of the techniques presented herein will be described with reference to software patches for an operating system (OS).

Aspects of the techniques presented herein support a method for recommending only the most relevant items to customers, or as candidates for automated deployment methods. Such ranking of the items ensures that only the highly relevant ones that are deployed in the largest and most active customers will be identified for deployment in customers that may be challenged (by, for example, a lack of skills, a lack of resources, etc.) in keeping up with software patch levels.

It is important to note that software patching is only one example of how aspects of the techniques presented herein may be employed. Further aspects of the techniques presented herein may utilize routing protocol features. For example, perhaps most customers who run Open Shortest Path First (OSPF) (i.e., Feature A) on a particular device (i.e., Feature B) choose to change the way OSPF metrics are calculated (i.e., Feature C).

In addition, the rate of enabling the new feature C once it became available is relatively high (thus increasing the relevance). This creates a strong recommendation to enable this feature for customers that have not yet enabled it.

The framework that is associated with the techniques presented herein may be applied to any feature recommendation, configuration best practice, security advisory remediation, etc. For simplicity of exposition the discussion that follows will focus on a software patching scenario.

Aspects of the techniques presented herein include a series of stages to create a ranked list of recommendations. For the purposes of illustration, in the example that is presented below various of the stages are highlighted using a very common yet complex scenario for recommendations – i.e., a base software that has both patch and service pack capability. As will be readily apparent, of particular interest and note are stages one and two (e.g., leveraging off-the-shelf solutions for feature identification) and stages three and four (e.g., developing a score for each feature based on a rate of adoption and continued usage by peers).

A first stage of the techniques presented herein encompasses a pairwise recommender system that is built using off-the-shelf algorithms. Pairwise feature comparisons with historical software patch data and lift metrics are used to provide an initial ranking of software patches, as well as an indication of which software patches are "real" because they have seen in real network data. For the general recommendations case, this could be classes of recommendations that have been actioned by other customers.

A second stage of the techniques presented herein employs the initial rate of deployment to combine with the lift and support metrics from stage one. This is a second scoring method which adjusts the patch ranking to account for deployment statistics seen across thousands of other network devices. Again, for the general recommendations case this would be the relative pace of taking action once recommendations were first shown to a customer (e.g., was it immediate, thus having a high rank, or did it linger, thus having a lower rank).

A third stage of the techniques presented herein utilizes network controllers or individual devices to collect the latest configuration data from unseen, new devices, and apply the ranked rules from stages one and two to generate a ranked list of candidate patches for each entry. In the general recommendations case, this could be the current "network improvement plan" or "action list" from a cloud service.

A fourth stage of the techniques presented herein is a filter stage that looks at user-to-user similarity (as opposed to item-to-item similarity in stages one and two) to determine the applicability of the recommended patches. One example is configuration similarity (e.g., modeled features as a configuration representation) to determine possible usefulness based on configurations of the device population that has the software patch installed. In the general recommendations case, this could be an industry peer group.

A fifth stage of the techniques presented herein integrates with an Information Technology Service Management (ITSM) system to open a ticket that includes, possibly among other things, the recommendation, the benefits to customers, and an option/services to make the change for the customer. This stage may also contain a final "don't recommend" list that is generated by expert systems matching (e.g., a patch is not relevant, a patch is in an installed service pack, etc.) as well as the input from stage six (e.g., a customer indicated no interest in the patch).

A sixth stage of the techniques presented herein closes the loop by developing a deployment plan (e.g., manual, automated system, etc.) or removing items that are no longer relevant (e.g., based on information from stage five) and then applying the changes in the network via the appropriate process for the type of recommendation.

Aspects of the narrative that was presented above may be illustrated through Figure 1, below.

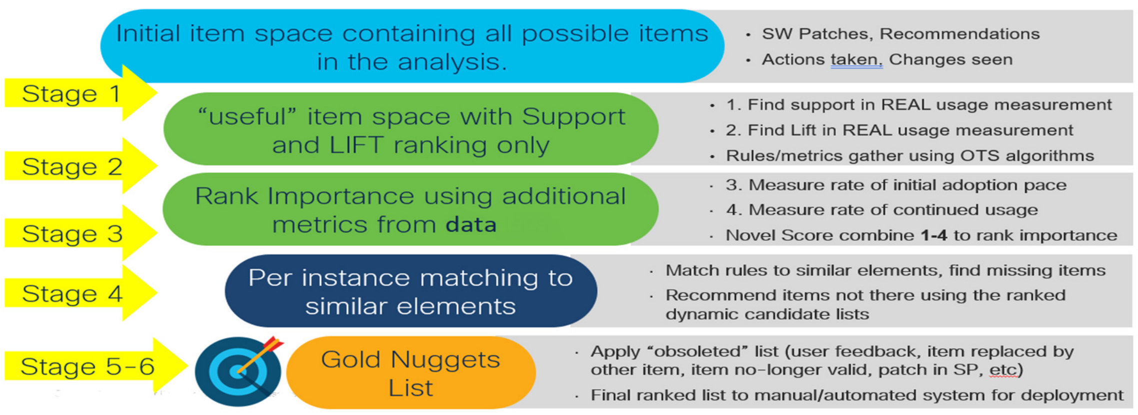

Initial item space containing all possible items in the analysis.
SW Patches, Recommendations
Actions taken, Changes seen
Stage 1
"useful" item space with Support and LIFT ranking only
1. Find support in REAL usage measurement
2. Find Lift in REAL usage measurement
Rules/metrics gather using OTS algorithms
Stage 2
Rank Importance using additional metrics from data
3. Measure rate of initial adoption pace
4. Measure rate of continued usage
Novel Score combine 1-4 to rank importance
Stage 3
Per instance matching to similar elements
Match rules to similar elements, find missing items
Recommend items not there using the ranked dynamic candidate lists
Stage 4
Stage 5-6
Gold Nuggets List
Apply "obsoleted" list (user feedback, item replaced by other item, item no-longer valid, patch in SP, etc)
Final ranked list to manual/automated system for deployment

Figure 1: Fasttrack Recommender System

As described above, stage one encompasses a pairwise recommender system that looks at all of the items in the item sets as pairs. This identifies items that are seen together, and can also double as a reasoner to determine that a patch is contained in other items. For example, as illustrated in Figure 2, below, both patches are contained in the service pack three. In any case of the customer having one but not the other, the system can infer that a service pack is not installed, and the missing feature should be recommended.

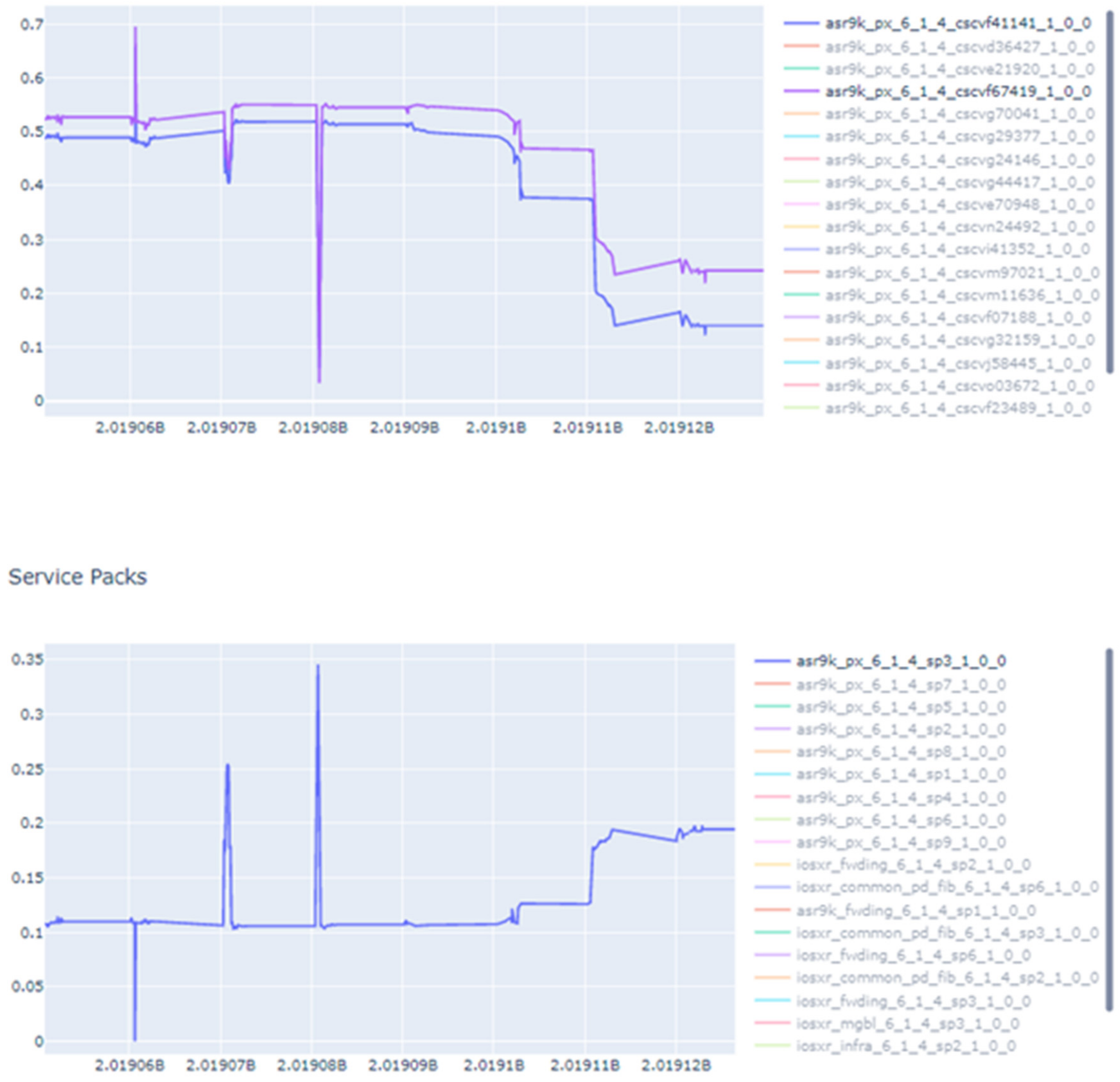


Figure 2: Illustrative Associations

In Figure 2, above, the top graph shows a clear relationship between two items commonly seen together, in a diagram that shows the overall deployment percentage. Any standard recommender system will pick these up as association rules. (Note that the Apriori algorithm was used in this instance.)

Figure 3, below, shows another example of patches that show up pairwise together in a different version of software. There are many patches covered, but there are clearly four standouts that are deployed at a much higher rate than any others. Those standouts get scored higher in stage two of the system. From some level of minimum deployment seen in real networks (such as, for example, 20% of the peer devices have the feature), an evaluation of the initial deployment line is examined and a score is generated. The purpose of the score is to indicate how fast the patch was deployed once it was made publicly available. Such a rate may be a proxy indicator that the patch is valuable and useful to most customers. In Figure 3, below, of note are the patches and the deployed rate observed after they were introduced.

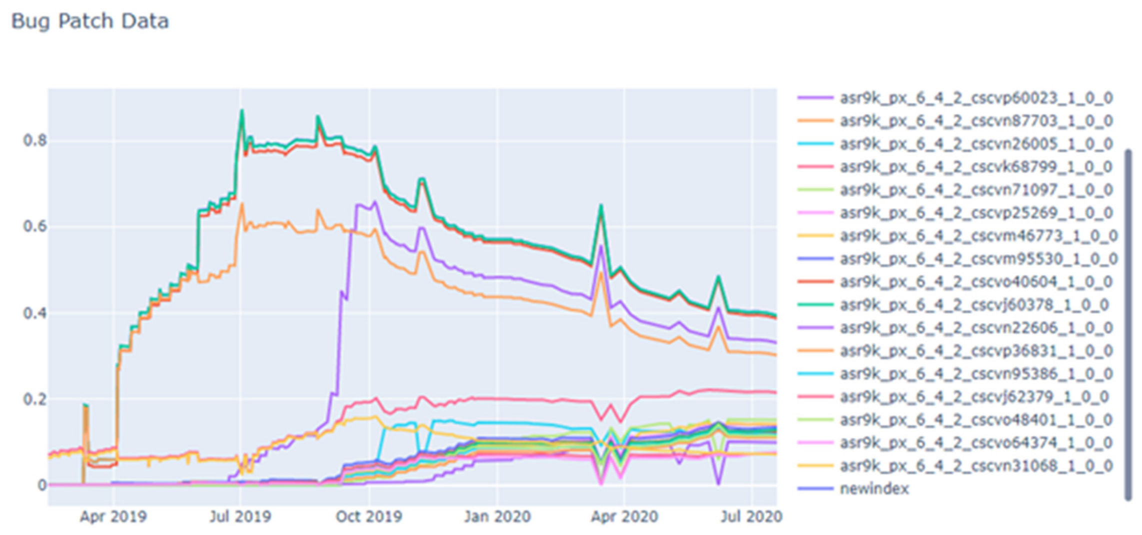


Figure 3: Patch Deployment Patterns

Figure 4, below, shows that there are six patches that are appearing in customer networks at a deployment rate greater than 20%. By evaluating the ratio of the number of days it took to reach maximum deployment and the actual maximum deployment, and enhancing that ratio with the number of days the patch had support over the interesting value, a relative score may be developed for each patch of interest using the following calculation:

P = Deployment percent maximum.

D = Days to go from a minimum to a maximum deployed percent.

L = Length of days the patch was above the interesting deployment rate.

P/D * L = A score used to rank the feature and influence the recommendation from stage one.

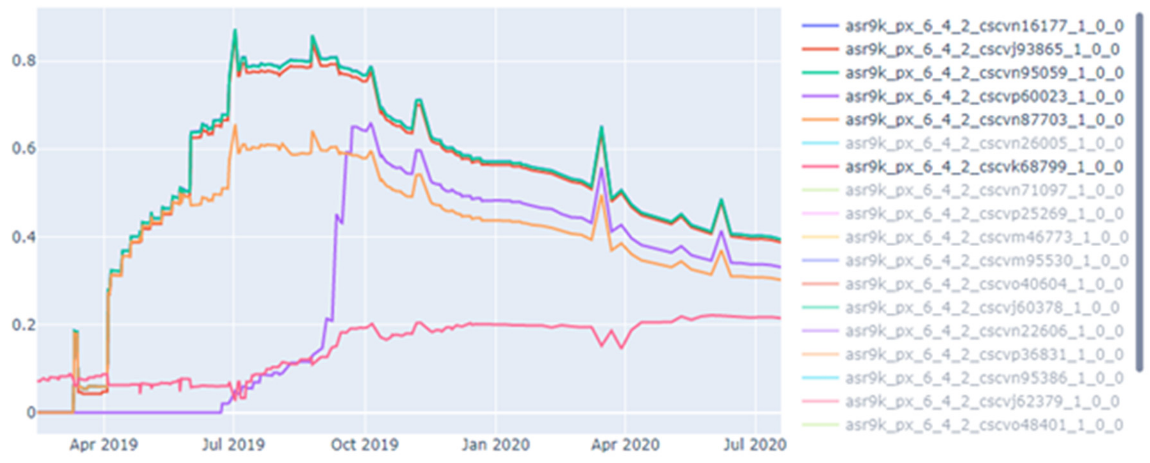
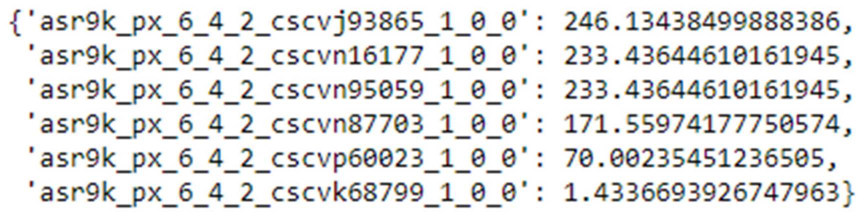


Figure 4: Relative Patch Deployments

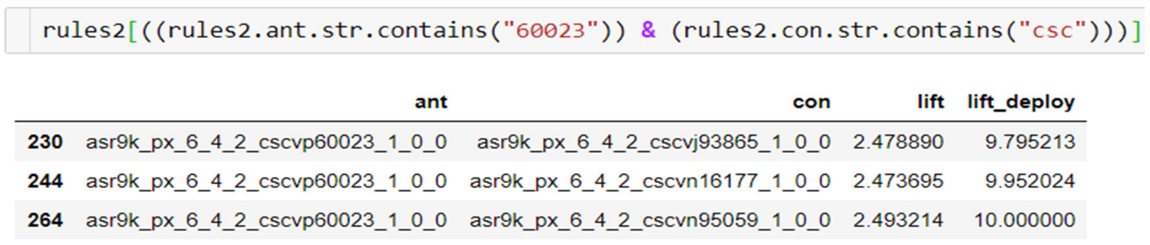
An example of the scoring algorithm for the features that were described above is shown in Figure 5, below.



```
{'asr9k_px_6_4_2_cscvj93865_1_0_0': 246.13438499888386,
 'asr9k_px_6_4_2_cscvn16177_1_0_0': 233.43644610161945,
 'asr9k_px_6_4_2_cscvn95059_1_0_0': 233.43644610161945,
 'asr9k_px_6_4_2_cscvn87703_1_0_0': 171.55974177750574,
 'asr9k_px_6_4_2_cscvp60023_1_0_0': 70.00235451236505,
 'asr9k_px_6_4_2_cscvk68799_1_0_0': 1.4336693926747963}
```

Figure 5: Illustrative Scoring

Under aspects of the techniques presented herein a score is normalized to fit within a range, such as, for example, zero (0) to ten (10). When all of the scores are evaluated together elements having very high deployment rates are identified and scored higher, adding quality to lift scores (e.g., from the off-the-shelf algorithms) that are relatively the same. As depicted in Figure 6, below, patch cscvn95059 is scored the highest and it is clearly visible as the most widely deployed patch in Figure 4, above.

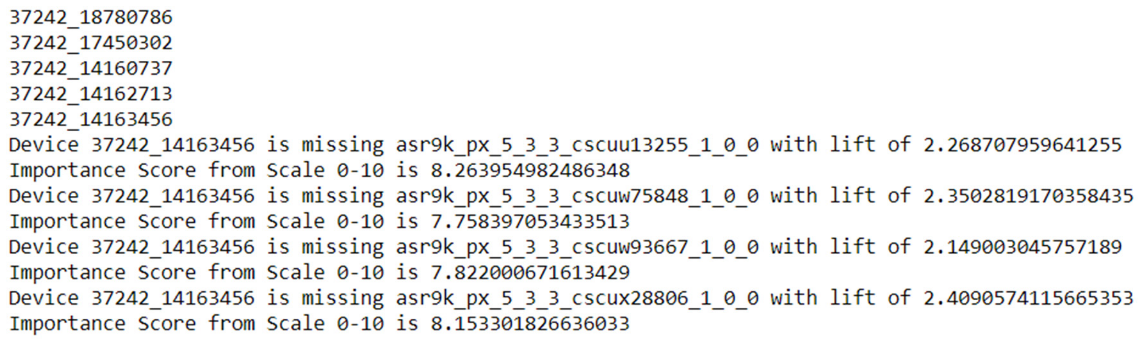


```
rules2[((rules2.ant.str.contains("60023")) & (rules2.con.str.contains("csc")))]
```

	ant	con	lift	lift_deploy
230	asr9k_px_6_4_2_cscvp60023_1_0_0	asr9k_px_6_4_2_cscvj93865_1_0_0	2.478890	9.795213
244	asr9k_px_6_4_2_cscvp60023_1_0_0	asr9k_px_6_4_2_cscvn16177_1_0_0	2.473695	9.952024
264	asr9k_px_6_4_2_cscvp60023_1_0_0	asr9k_px_6_4_2_cscvn95059_1_0_0	2.493214	10.000000

Figure 6: Normalized Scoring

During stage four, after similar devices are selected the devices are searched for antecedents and if a consequent is not found it is recommended. All of the scoring is provided for ranking via any system that will ingest same and provide a presentation layer. Figure 7, below, depicts an example of applying the rules and generating the recommendations and scores that would be output. Note that each underscored item is a real customer device.



```
37242_18780786
37242_17450302
37242_14160737
37242_14162713
37242_14163456
Device 37242_14163456 is missing asr9k_px_5_3_3_cscuu13255_1_0_0 with lift of 2.268707959641255
Importance Score from Scale 0-10 is 8.263954982486348
Device 37242_14163456 is missing asr9k_px_5_3_3_cscuw75848_1_0_0 with lift of 2.3502819170358435
Importance Score from Scale 0-10 is 7.758397053433513
Device 37242_14163456 is missing asr9k_px_5_3_3_cscuw93667_1_0_0 with lift of 2.149003045757189
Importance Score from Scale 0-10 is 7.822000671613429
Device 37242_14163456 is missing asr9k_px_5_3_3_cscux28806_1_0_0 with lift of 2.4090574115665353
Importance Score from Scale 0-10 is 8.153301826636033
```

Figure 7: Illustrative Recommendations and Scores

It is important to note that a key element for a useful system is the "null" list that is developed during stage five. For a variety of reasons, recommendations that arise from the techniques presented herein (or from any system like it) may be applicable in a technical sense but may not be of interest to an end user. This list may be maintained using inputs from all possible sources, including, for example, user inputs, cross validations (e.g., perhaps one item supersedes another, etc.), or any other methods.

It is also important to note that the techniques presented herein do not replace any full software upgrade systems, such as, for example, Optimal Software Versions (OSV), but instead enhances cases where a full software upgrade is not necessary because patches are available.

Overall, the techniques that are presented herein can recommend changes that are most applicable as part of, for example, risk remediation platforms including configuration best practices, software updates (e.g., patching), security advisories, feature adoption, and more, in order to increase the value, context, and adoption success of recommendations to customers.

In summary, techniques have been presented that provide a novel approach to generating and ranking recommendations coming from a dynamic recommender system where rankings are based on enriched context from, for example, live data on real networks, activities performed by real customers, etc. Such techniques enhance the operational features of existing networks by recommending popular items to new customers, identifying critical items that can be proactively addressed in order to provide additional services, and reducing TAC cases when patches exist for common issues. In some instances, the techniques may facilitate using an off the shelf recommender system for features combined with a score for each feature based on rate of adoption and continued usage by peers.