InBeat: News Recommender System as a Service @ CLEF-NEWSREEL'14

Jaroslav Kuchař[1,2] and Tomáš Kliegr[2,3]

[1] Web Engineering Group, Faculty of Information Technology,
Czech Technical University in Prague
`firstname.lastname@fit.cvut.cz`
[2] Dep. of Information and Knowledge Engineering,
Faculty of Informatics and Statistics, University of Economics Prague
`firstname.lastname@vse.cz`
[3] Multimedia and Vision Research Group
Queen Mary, University of London

Abstract. Interest Beat (`inbeat.eu`) is a service for recommendation of content. InBeat was designed with emphasis on versatility, scalability and extensibility. The core contains the *General Analytics INterceptor* module, which collects and aggregates user interactions, the *Preference Learning module* and the *Recommender System* module. In this paper, we describe InBeat general architecture, putting emphasis on its high-performance architecture that was used in the CLEF-NEWSREEL: News Recommendation Evaluation Lab.

Keywords: recommender system, web service, architecture, preference learning, scalability, challenge

1 Introduction

The large amount of content to choose from causes the *Information Overload* problem for visitors of news websites. Based on the analysis of past usage patterns, recommender systems can make a personalized list of preselected content, alleviating the users of the effort entailed in the process of choosing the content they should consume next and limiting the number of choices they need to make.

There are various existing recommendation systems and algorithms, both in academia and industry.While recommender systems were used on large e-commerce websites for more than a decade [6], there is still a limited range of *generic* recommender systems that are deployable with little or no customization. With InBeat, we aim at providing a ready-made and versatile recommender for high-throughput use cases. InBeat provides RESTful services for all inputs and outputs, which streamlines its integration with any Internet-connected application supporting the HTTP protocol.

In this paper, we introduce the InBeat architecture and algorithms in the context of its participation in the CLEF-NEWSREEL: News Recommendation Evaluation Lab (further only Challenge), which focused on recommending news

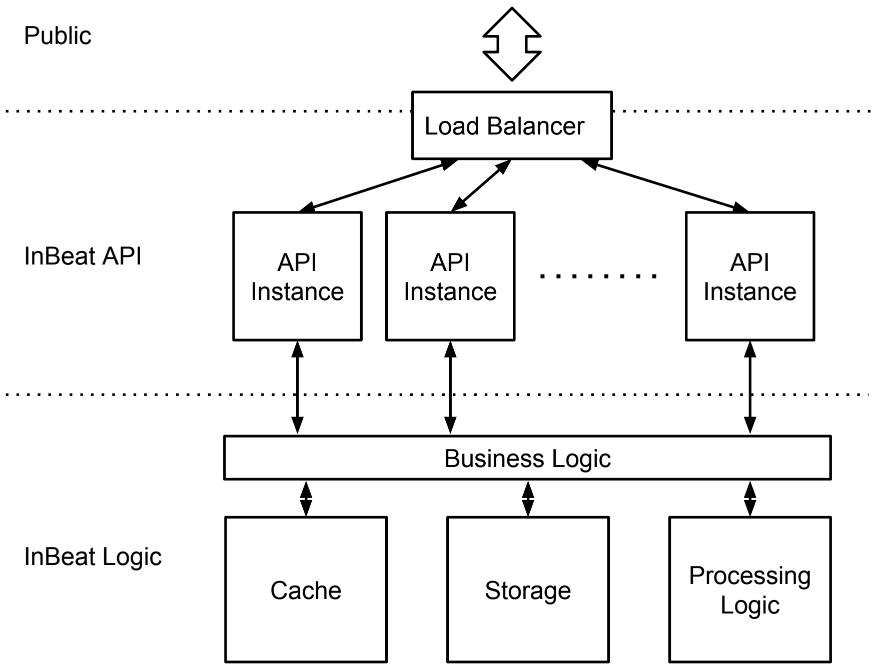


Fig. 1. InBeat scalable solution

articles in real-time.In section 2 presents the system architecture and technical solution. Section 3 presents InBeat as a news recommender system in the Challenge's on-line task, including the details on performance. Section 4 presents our attempt at the off-line task. Finally, Section 5 provides a list of other InBeat use cases along with some reflections on the organization of the Challenge.

2 Technical Solution

InBeat's was designed for scalability and low utilization of resources per one client. A simplified overview of the technical solution is given in Figure 1. The main entry point is load balancer *nginx*[4], which distributes the workloads across multiple instances of API applications. Nginx is focused on high concurrency, performance and low memory usage. It uses asynchronous event-driven approach and is able to provide high performance under high loads.

Instances are created with *Node.js*[5], which is a platform for building fast and scalable applications. Node.js uses an event-driven, non-blocking I/O model that

[4] `http://nginx.org/`
[5] `http://nodejs.org/`

makes it lightweight and efficient, suitable for data-intensive real-time applications that run across distributed devices. As storage we selected *MongoDB*[6], which met our demands on scalability, performance and high availability. Its key advantage is schema-less design, which allows to use custom set of attributes in an individual use case without the need to update the schema. MongoDB also supports map/reduce procedures that are used for aggregation of data and for providing different views on data by InBeat. We use MongoDB mainly as a storage of historic data (e.g. latest N thousands interactions per news portal or descriptions of items) and aggregated statistics (e.g. number of interactions per item etc.). The Cache is implemented with *Redis*[7], a highly scalable in-memory, key-value store. It provides essential performance improvement under high workloads, reducing the latency of business logic including storage operations.

3 On-line task: Setup and Results

The CLEF-NEWSREEL: News Recommendation Evaluation Lab[8] is focused on recommending news articles in real-time. The emphasis of the Challenge is on scalability and response time limitation. Recommendations had to be provided in real-time (within 100 ms), and the winning criterion was set to the total number of successful recommendations, rather than the prediction accuracy (clickthrough rate). There are practical problems with real time processing of recommendations that are not incurred when there is "unlimited time" to provide the recommendation. It is necessary to balance the architecture and technologies with the complexity of the involved algorithms. The InBeat platform deals with real time processing of data using special features of NoSQL databases and map/reduce principles.

3.1 Task Definition

This section describes simplified definition of the news recommender task.
Inputs: The main inputs are the users' interactions and news item descriptions.

- $interaction(type, userId, itemId, context)$
  where $type = \{impression|click\}$ and *context* describes the features of the user (e.g. browser version, geolocation, etc.) and special features related to items and their presentation (e.g. keywords, position).
- $item(itemId, domain, description)$
  where *domain* is the identifier of items from the same group (e.g. news portal) and *desciption* provides more detailed information about items (e.g. title, text, time of last update).

Outputs: Set of recommended items for the specific user who is reading the item within a given context.

- $(userId, itemId, context) \rightarrow \{item_x, item_y, ...\}$

[6] http://www.mongodb.org/
[7] http://redis.io/
[8] http://www.clef-newsreel.org/

3.2 Algorithms

In this section we describe a set of algorithms we used in the Challenge.

Top Interacted This algorithm is based on the daily popularity of news items. To avoid excessive effect of high short-time popularity of one item the interactions are aggregated on a daily basis. This approach deals with an evolution of popularity over time and decrease an influence of peaks appeared at the specific days. We implemented the algorithm using simple incremental updates in a MongoDB collection represented as a triple $(Date, ItemId, count)$. The result is s list of items sorted by the number of interactions.

Most Recent Since we are in the highly dynamic news domain, the recency of an article plays an important. Our baseline recency-based algorithm uses a simple heuristic based on the newest news item within the same group as the group of the item the user is reading at the time of the request. The results is ordered list of items sorted by creation time.

Rule Based Since this algorithm is more complex than the previous one, we decided to describe it in greater detail.

Table 1. Training dataset for rule based recommender.

Context							Class
browser	isp	os	geo	weekday	lang	zip	item
312613	281	431229	19051	26887	49021	62015	127563250
457399	45	952253	18851	26887	48985	65537	45360072

For each $interaction(type, userId, itemId, context)$ stored in our database, we prepared one entry in the training dataset as described in Table 1. Interactions are described only by the contextual features that are provided by the platform (e.g. Location, Browser, ...) and by an identifier of item the user interacted with.

The training dataset was used to learn association rules. The contextual features could appear only in the rule body (antecedent) and the identifier of the item only on the right side of rule (consequent). We used association mining algorithm *apriori* implemented in *R - arules*[9]. Example of a rule:

$$\text{isp} = \text{“281”} \ \wedge \ \text{os} = \text{“431229”} \ \rightarrow \ \text{item} = \text{“1124541”}$$

Additional mining setup is as follows. We used latest N thousands interactions as training dataset from our database. We experimentally set N to five thousands. The apriori algorithm is experimentally constrained with minimal support of five interactions, and minimum confidence of 0.2.

[9] http://cran.r-project.org/web/packages/arules/

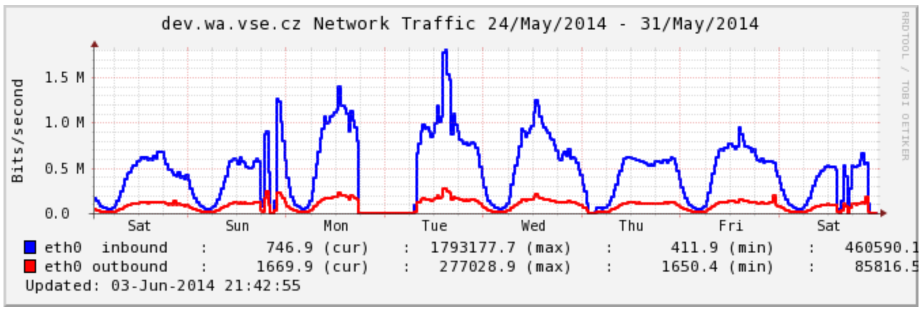


Fig. 2. Network Traffic on InBeat server.

All discovered rules are imported into our simple rule engine. The engine finds all rules that match the contextual features of a recommendation request. The consequent of each matching rule represents a recommended item. The output is a list of unique item identifiers from the right side of the matching rules.

The association rule-based algorithm was added at the end of Challenge and thus did not participate in all evaluation weeks.

3.3 Performance

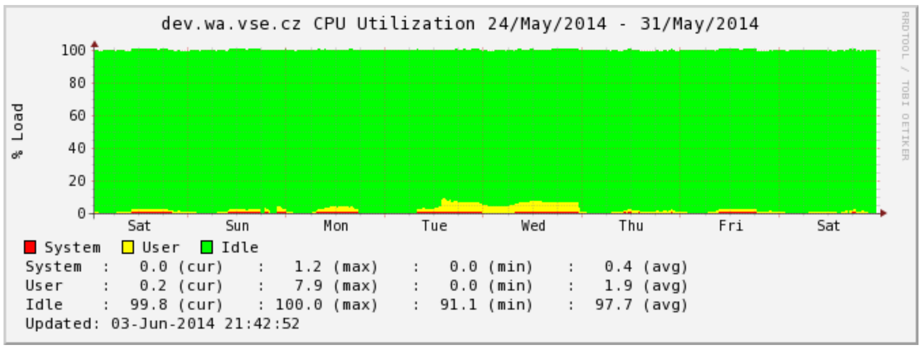
In this section, we present the performance of the InBeat recommender in the Challenge. The metrics used in the Challenge to select the winning recommender systems was the *cumulative number of clicks* (number of successful recommendations) over the three different evaluation periods. The additional metrics provided by the organizers include *number of impressions* and *click-through rate*.

Sum of the number of impressions with the number of clicks can be interpreted as the performance of the systems – the ability to process large number of interaction on the server.

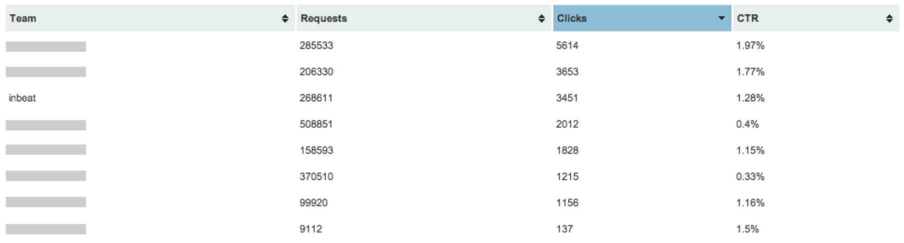
Figure 2 shows the network traffic on our server infrastructure within the last week of the Challenge. During this period, InBeat handled thousands of recommendation requests. The peaks in the graph correspond to the higher number of interaction in daytime. Note that the gap between Monday and Tuesday is caused by the maintenance break of our infrastructure. Figure 3 depicts the CPU load on the server. The server load was kept mostly under ten percent even in peak periods. InBeat was run on a single virtual machine assigned four Core i7@3.20GHz cores and 8GB of RAM.

Figure 4 presents the results for the last evaluation period. The table is sorted by the cumulative number of clicks. InBeat team is on the third position. The table provides only results that are aggregated per team participating in the Challenge. There are no specific results for each recommendation algorithm. In click-through rate, the second metric, InBeat is on the fourth position.

Since the CTR reported in Figure 4 is the average for all algorithms, also report the numbers for the individual InBeat algorithms:



```
System  :   0.0 (cur)   :   1.2 (max)   :   0.0 (min)   :   0.4 (avg)
User    :   0.2 (cur)   :   7.9 (max)   :   0.0 (min)   :   1.9 (avg)
Idle    :  99.8 (cur)   : 100.0 (max)   :  91.1 (min)   :  97.7 (avg)
Updated: 03-Jun-2014 21:42:52
```



Team	Requests	Clicks	CTR
[redacted]	285533	5614	1.97%
[redacted]	206330	3653	1.77%
inbeat	268611	3451	1.28%
[redacted]	508851	2012	0.4%
[redacted]	158593	1828	1.15%
[redacted]	370510	1215	0.33%
[redacted]	99920	1156	1.16%
[redacted]	9112	137	1.5%

Fig. 4. Leaderboard with cumulative number of clicks and average click-through rate per team in Challenge - last evaluation period 2014-05-25 - 2014-05-31. Source: http://orp.plista.com

- *Top Interacted* has 1.4% CTR,
- *Most Recent* has 0.8% CTR,
- *Rule Based* has 1.5% CTR.

The most successful algorithm is *Rule Based*, which we explain by the fact that it takes into account both popularity and contextual features. *Most Recent* is influenced only by temporal aspects and *Top Interacted* takes into account only the popularity.

4 Off-line task: Setup and Results

In addition to the on-line challenge CLEF-NEWSREEL offered an off-line task. The dataset consisted of 84 million records of recommendation data across multiple portals [1].

4.1 Task definition

From the off-line dataset, we used a subset of 26,875 records.[10] The data were preprocessed to the form shown at Table 1 and randomly split to a training

[10] From the portal with the smallest number of records.

dataset (90%) and test dataset (10%). The task was to predict the label (item viewed), given the description of the context. In our approach, we have not exploited the temporal relations between entries in the training dataset. Each row was considered as a separate training instance.

4.2 Algorithms

We have used several standard approaches as a baseline, and compared it with the proprietary InBeat rule learning solution.

Baseline: Decision trees, SVM. We used the implementations of common machine learning algorithms available in RapidMiner 5: Decision trees (CHAID and the RapidMiner's "Decision Tree"), and SVM (LibSVM).

Rule learning – basic. R-arules implementation of the apriori algorithm, in the fastest setup described in Section 3. The resulting rule set is sorted according to confidence, support, and rule length (shorter is ranked better/higher). The top ranked rule matching a test object is used for classification.

Rule learning – with pruning. R-arules implementation of the apriori algorithm. After learning, the rules are pruned using our database coverage pruning implementation [4]. The rules are matched with test data using the same approach as in the basic setup. This setup is a simplification of the seminal CBA (Classification Based on Associations) algorithm.

Rule learning – pruning with extension. Same as the previous setup, but the rules in the pruned rule set are additionally post-processed using *rule extension* (working version of our experimental algorithm). The rules are matched with test data using the same approach as in the basic setup.

Rule learning – pruning with extension and mixture classification. Same as the previous setup, but all rules matching the test object contribute to the final classification using the *rule mixture* approach (a working version of our experimental algorithm).

4.3 Performance

The results are depicted on Table 2. The baseline algorithms were run with default parameters in the RapidMiner environment.[11] Rule learning was run with 0.02 minimum confidence and 0.001 minimum relative support threshold.

The worst results were provided by CHAID and Decision Tree algorithms, the best performing result was provided by SVM (with RBF kernel).

[11] For CHAID and DecTree we tried several different parameter configurations, but with no or negligible improvement.

Table 2. Training dataset for rule based recommender.

	SVM	CHAID	DecTree	RL-Basic	RL-PR	RL-PR-Ext	RL-PR-Ext-Mix
Accuracy	0.15	0.02	0.02	0.02	0.06	0.08	0.11

The results clearly indicate that while our experimental rule-based system surpassed other symbolic learning algorithms incl. CHAID and Decision Tree, it was not able to beat the SVM with RBF kernel baseline.

5 Conclusion and Future Work

In this paper we introduced the *InBeat* experimental recommender system. There is an on-going work on extending the set of recommendation algorithms available in InBeat and their customization to the constraints imposed by the Challenge platform, which remains open for further experiments.

It should be noted that InBeat or its modules are or were deployed also in other use cases. The first version of its component was conceived as a web analytics solution compatible with Google Analytics [2] and was deployed on a travel agency website. The option to process web analytics data was extended in the current version with support for the recent version of the Google Analytics tracking code `ga.js`. InBeat is currently primarily extended as a component of a "SMART-TV" recommender system [5]. The most recent development in this direction is the ability to process feedback from user-behaviour tracking with Microsoft Kinect [3].

Acknowledgement. This work was supported by the Czech Technical University grant SGS14/104/OHK3/1T/18, by the University of Economics, Prague by grant IGA 26/2011 and by the EC project FP7-287911 LinkedTV.

References

1. B. Kille, F. Hopfgartner, T. Brodt, and T. Heintz. The plista dataset. In *NRS'13: Proceedings of the International Workshop and Challenge on News Recommender Systems*, ICPS, p. 14–22. ACM, 10 2013.
2. T. Kliegr. *Clickstream Analysis.* University of Economics in Prague, Faculty of Informatics and Statistics, Prague, 2007. Master Thesis.
3. T. Kliegr and J. Kuchař. Orwellian Eye: Video recommendation with Microsoft Kinect. In *In Conference on Prestigious Applications of Intelligent Systems (PAIS'14) collocated with European Conference on Artificial Intelligence (ECAI'14)*. IOS Press, August 2014. To appear.
4. T. Kliegr, J. Kuchař, D. Sottara, and S. Vojíř. Learning business rules with association rule classifiers. In *RuleML*. 2014. To appear.
5. J. Kuchař and T. Kliegr. GAIN: web service for user tracking and preference learning - a SMART TV use case. In *7th ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*. 2013.
6. J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in E-commerce. In *Proceedings of the 1st ACM Conference on Electronic commerce*, EC '99, pp. 158–166. ACM, New York, NY, USA, 1999.