PolyRecs: Improving Page–View Rates Using Real-Time Data Analysis

Mihalis Papakonstantinou and **Alex Delis**

Univ. of Athens, Athens 15703, Greece
{mihalispapak,ad}@di.uoa.gr

Abstract. In this paper, we outline our effort to enhance the page-view rates of *e*-content that online customers read on a popular portal in Greece. The portal, athensvoice.gr, provides continuous coverage on news, politics, science, the arts, and opinion columns and its customers generate approximately 6 million unique visits per month. Gains both in terms of advertisement and further e-content market penetration were the objectives of our effort which yielded the PolyRecs system, in production for more than a year now. In designing PolyRecs, we were primarily concerned with the use of pages in real-time and to this end, we elected to utilize five key criteria to achieve the aforementioned goals. We selected criteria for which we were able to obtain pertinent statistics without compromising performance and offered a real-time exploitation of the user page-views on the go. In addition, we were keen in realizing not only effective on-the-fly calculations of what might be interesting to the browsing individuals at specific points in time but also produce accurate results capable of improving the user-experience. The key factors exploited by PolyRecs entail features from both collaboration and content-based systems. Once operational, PolyRecs helped the news portal attain an average increase of 6.3% of the overall page-views in its traffic. To ascertain the PolyRecs utility, we provide a brief economic analysis in terms of measured performance indicators and identify the degree of contribution each of the key factors offers. Last but not least, we have developed PolyRecs as a domain-agnostic hybrid-recommendation system for we wanted it to successfully function regardless of the underlying data and/or content infrastructure.

Key words: timely delivery of news articles, content-based furnishing of news, hybrid recommendation systems, real-time content analysis.

1 Introduction

Serving just-on-time pertinent news items has become a key requirement for a broad range of web portals, online publications and news agencies [1, 2]. Appropriately recommending content in real-time has become the enabler for conducting successful business in both online dissemination and e-publishing. By being proactive and by furnishing more accurate suggestions providers undoubtedly benefit as users expend longer periods of time online, click-through-rates are increased and ultimately, revenue grows either directly from advertisements or indirectly from ratings [3, 4].

Until recently, suggestions for further reading for no-global content providers have been mostly served using either one or a combination of the following criteria: *a)* time that an article appeared, *b)* selection of articles from a single subject category, *c)* choosing items either with semi- or manual manner, and *d)* picking items written by the same author, source, etc. In this paper, we outline an effort to *enhance* the existing recommendation system of `athensvoice.gr`, a popular news portal in the country that offers news, articles on politics, arts, local affairs, opinion columns and user comments; the portal has been operational for a number of years now and has seen its market share to increase. We seek to accomplish the aforementioned objective by offering a novel combined approach of content-based and collaborative filtering [5] along with real-time analysis for newly arriving news items and improved user-experience through more effective suggestions of items. To this end, we have improved the operation of the *e-magazine* by offering a truly hybrid system for recommendations. To the best of our knowledge, this is the first system in its kind that is both hybrid in nature and deploys multiple selection criteria that get to be evaluated in real-time.

An exploratory effort to better understand the specific constraints based on *JavaScript*-functions used to provide workbench measurements for `PolyRecs`'s key functionalities showed that the slack for the time window within which recommendation can be compiled is approximately 1 second before the user actually sees the corresponding portion of the display. This is clearly a very tight period within which both computation and displaying have to occur. We should indicate that `athensvoice.gr` receives approximately 6 million visitors from unique `IP`s every month, with an average of 2.3 pages served per user.

The developed system, termed `PolyRecs`, has both real-time and off–line components. It serves recommendations produced dynamically as a visitor scrolls within a web-page in a pre-specified portion of the page; the form of latter differs across devices used to access content from the portal. Our prime objective has been two-pronged as we seek to: *1)* maintain strict requirements for the system's responsiveness by having 97% of suggestions appearing in less than 1 second; displaying suggestions does call for computing overheads that include elimination of news items already seen, incorporation of just-arrived-pieces in the display, and implications due to specific profiling users may elect to adopt, *2)* divide the computational work as effective as possible between the off–line (initial) component of the portal and the introduced on-line part of `PolyRecs`. The off–line work gets done mostly through `cron` script jobs and they mostly involve computations that have to do with the re-computation of correlation among subject categories. In particular the tasks at hand entail computing cosine similarities among new and old articles, backing up data and purging obsolete ones no longer required for the engine, and finally, clearing up data from the database table that are considered noise [6, 7]. This task is a CPU-hog [8] but in short time intervals required by the average portal user, it contributes to the requisite overhead only incrementally.

We present the specific multiple criteria that collectively contribute towards recommendations presented to unique users and outline the `PolyRecs` functional elements that place special emphasis on new posts. Clearly, the corpus of diversified content items for `athensvoice.gr` does continuously change and/or gets updated. It is worth mentioning that we have designed and developed both the underlying database and content delivery system in a way that is agnostic to the particular characteristics of the portal for which we sought out to increase page-views. In this regard, the salient features of `PolyRecs` recommendation engine could be readily incorporated into news and dissemination portals.

PolyRecs has been operational for more than an year now and this has given us the opportunity to carry out extensive experimentation and evaluation of all its key aspects. To gauge the yield in terms of economic benefit to the site, we introduce a metric to quantify possible gains; over time, the metric has helped us ascertain the effectiveness of our overall recommendation strategy in conjunction with the data we obtain from the *Google Analytics Service*. The proposed metric uses a visitor's clicks as its main factor that helps track the criteria which triggered the produced recommendations.

Our approach has assisted athensvoice.gr to increase on the average the time visitors stay on the portal by approximately 6% as well as it has yielded improved rates for content retrieval. To the best of our knowledge, this is the first such production system, that is both hybrid and uses multiple criteria, deployed and evaluated on an operational *e*-content portal. Our paper is organized as follows: Section 2 offers related work and Section 3 outlines key requirements for the proposed system. Section 4 presents all key aspects of our recommendation approach. Finally, Sections 5 and 6 presents key findings and concluding remarks respectively.

2 Related Work

The emergence of digital forums, e-shops, and modern electronic marketing ecosystems has ushered in a flurry of activity in developing recommending systems [1, 5, 3, 9, 10]. Moreover, a range of metrics have been proposed to help ascertain the corresponding value of such prototypes and/or production systems in specific areas of application [11, 12, 13, 14, 4, 5, 6].

Efforts in [5, 1] have focused on recommendations for search engines using machine learning algorithms for predicting news on the wire [15, 16], expanded vertical search [17] and crowd-sourcing to better evaluate the learned approaches deployed [1]. In [3] the deployment of a hybrid recommending engine for the Google-News, a news aggregator, and its comparison with a collaborative filtering approach [9] is carried out; if only logged–in users are considered, a noteworthy performance improvement is reported with respect to plain filtering approaches. In [18], the imprecision of the click-through rates is examined in the context of the *Plista* news-recommender on hourly and weekly bases; *Plista* functions as an aggregator and delivers suggestions to multiple web portals. Our approach combines characteristics from both collaborative [2] and item/user-based content-based filtering [5, 3]. In this respect, our approach resembles in part prior efforts [5, 11, 3] when it comes to combining techniques for yielding recommendations. However, we use diverse criteria that can be dynamically introduced during the calibration of our system. More importantly, we combine real-time and off–line characteristics for we strive to comply with short timing requirements to produce quality suggestions for users.

In [6, 19, 1], a number of metrics were introduced using predominantly click-rates to derive recommendations; the effectiveness of such metrics is also compared with that of prior approaches [6, 19, 1]. PolyRecs markedly differs from the above efforts as we not only exploit clicks users perform but more importantly, we keep track of the degree by which the used criteria are affected by the dwelling of unique users. This does lead to a more sophisticated calibration of the overall recommendation engine used by athensvoice.gr. Moreover, PolyRecs provides a framework for introducing specialized machine learning approaches in recommending items from the continually changing content of the portal.

3 PolyRecs Approach

As we wanted to improve PolyRecs page-views and increase unique visits to the production portal, we actively avoided during the design of the engine to provide recommendations based on widely used heuristics such as articles written by same (or similar) authors, content in the same category and/or just popular items within the portal. These techniques although effective at times, fail to consider a reader's own interests at the time of browsing. In addition, prior stated-preferences may be discarded and possible correlations among categories might be overlooked.

In PolyRecs, we combine a number of techniques so that we can present visitors with suggestions that not only fall within the realm of their interests but they also are "fresh"; the notion of being "fresh" pertains to pages that either have not been read so far or may have appeared on the portal data infrastructure recently. In this context, a key factor that we had to take into consideration was the very large amount of data collected within a matter of minutes in user activity. Effectively managing the inflow of information as well as the outflow of the portal data constitutes a challenge. In an exploratory phase, we instrumented athensvoice.gr and measured the average dwelling time for a visitor: we found it consistently to be around 1 *second* using *JavaScript* events. It is within this window of time that recommendations have to be compiled and be timely shown to the reader so as to increase page-view rates and consequently time spent on athensvoice.gr. This timing benchmark also indicated that regardless of the sophistication of the recommendation algorithm, if we take longer times to generate suggestions this will render the engine ineffective. In this case, the majority of visitors will fail to receive both personalized and accurate recommendations within the designated real-time slack. To accommodate the above requirement, PolyRecs's design follows a *hybrid* approach exploiting a variety of key operational aspects by:

1. predominantly focusing on the responsiveness of such an engine,
2. being able to process important data flows and events real-time,
3. profiling both users and incoming pieces of content in a timely manner,
4. deploying effective features from traditional techniques used in the athensvoice.gr engine to this date,
5. training on–the–fly as much as possible while relegating off–line work for resource-intensive tasks only,
6. designing a system that can work in a plug-and-play fashion, regardless of the underlying infrastructure (i.e., database and content management systems used).

All the above features influenced the design of PolyRecs and led us to deploy an engine that embeds multi-criteria in its core operation with the time slack for all jobs taken continuously into account. In addition, we deploy a fail-over mechanism that addresses issues introduced due to new content, users, content category re-alignment, and classification re-adjustment.

A number of factors influence the way PolyRecs yields its suggestions and include:

- the visitor's unique profile based on user-id and IP number,
- articles whose categories are strongly related to content currently in browsing,
- the time of day the visitor is browsing the portal,
- articles that share a high textual–similarity to the one currently being accessed,
- articles that are popular on this content–category today.

By default, each of the above criteria may equally contribute –in terms of weight– to the outcomes computed by PolyRecs. In this regard, we seek to ensure the objectiveness of

the evaluation while we offer the capability to appropriately gauge the weighting scheme so as to provide content of timely interest. Moreover, we want to offer warranties that PolyRecs regardless of the nature of the visitor and/or the item currently in browsing, we can locate suggestions even if one or more of the contributing criteria fail to produce suggestions. In this case, the rest of the criteria will kick in and help fill in the required quota for compiling the list of the suggestions.

We aspire to carry the necessary functionalities in real-time and compute all required aspects on-the-fly in a way that the user has the time to view recommendations, evaluate their worthiness and likely proceed to read one of those suggestions. If so, a new batch of recommendations is computed while user clicks generate valuable input in term of the navigation provided for our engine.

4 PolyRecs Architecture

This section outlines the overall functional architecture of PolyRecs in a way that mostly focuses in the flow of data as well as both user-input/output generated. Figure 1 depicts the overall organization in terms of constituent components of PolyRecs as well as the data/control interactions with the other major components of the athensvoice.gr. We realize the PolyRecs engine by weaving five key elements together:

1. the database system (*DBMS*) responsible for storing and querying user traffic events and requisite (meta-)data,
2. the recommendation engine (*RE*) responsible for processing the data and executing our recommendation algorithm,
3. the interface handler (*IH*) which appropriately displays results based on the devices users interact with,
4. the maintenance system (*MSys*) that performs all necessary off-line computation work and maintenance operations such as cleaning, staging and archiving,
5. the injector (*Inj*) that helps quantify the (financial) benefits of integrating PolyRecs to the athensvoice.gr by code-instrumenting the suggestions made.

Although PolyRecs is embedded in the athensvoice.gr portal and essentially interacts with the respective *Apache* web-server and *Drupal* CMS, we have followed a highly modular design so that it can function with any other portal layout. In what follows, we discuss the operation of the key PolyRecs components and indicate their interactions.

4.1 The Database (*DBMS*)

MySQL is deployed to help store data and realize queries in PolyRecs while using two schemas: the *main* schema that essentially hosts all real-time data and a *backup* schema in which we stage data deemed stale yet potentially useful for forthcoming retrieval(s). The more critical of the tables existing in the PolyRecs database main schema manages all data that are generated by data flows incurred by the user activity. The table stores facts such as user-ids involved in accessing portal content, time-stamps and URLs visited. Once acquired through the click-through activity of users, these data have to be further processed and passed along to other database tables holding respective information. In this context, every URL gets analyzed to its ingredients that are: the article-id (information shared between the DBMSs of both portal and PolyRecs), category classification, as well as time-stamp of the click. The main table termed *init_data*,

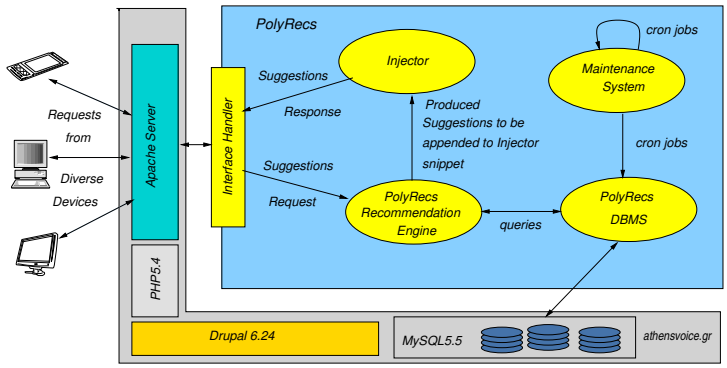


Fig. 1: `PolyRecs` Functional Overview in the context of *athensvoice.gr*

rapidly builds volume due to all recorded input/output operations performed on the portal's *Apache*-server on a daily basis. Apparently, the way we handle this table may affect the performance of our engine and is deemed an enabler for furnishing timely responses to user queries.

Overall, the main schema of our *MySQL* database features tables that predominantly store processed information accumulated over time, and statistics that are fed to `PolyRecs`'s recommendation engine to assist in creating suggestions. Among others, these tables include: 1) the users profile table, 2) the categories profile table, 3) day time profile table, 4) article to article similarity table, 5) popular articles per day table. We should point out that the information maintained by the above tables has to do mostly with meta-data of articles and items whose actual content resides in the portal's database. The design of the backup schema is identical to that of the main for its hosts dated, less useful or occasionally obsolete information and gets to be occasionally purged.

4.2 `PolyRecs` Recommendation Engine (*RE*)

The engine undertakes the task to produce within the time slack permitted, suggestions for a specific visitor. Its function highly depends on the database and essentially works as a go-between the interface and *MySQL*. Given as input the unique identifier for a visitor in conjunction with the current `URL` he/she is viewing (from which the article-id and category are derived), the engine returns a predefined number of suggestions. The input in discussion is used by the *RE* as follows: *a)* the visitor's id is utilized to fetch and enrich his/hers profile in terms of favorable categories, pieces of content already read, preferences, etc, *b)* the `URL` currently being viewed is marked as seen and is used to "select" a category of items so that similar items are suggested, and *c)* `PolyRecs` by design imposes an upper-limit on the number of suggestions displayed to its user interface; this can lighten the work carried out by the database in its effort to produce *top-k* items for queries on suggestions.

After a lengthy empirical evaluation and detailed study of user-cases, we set this k=15 for predominantly two reasons: i) the portion of the web-page reserved for these suggestions could not exceed this limit for this would not effectively facilitate the user view of the results, and ii) as we mainly derive recommendation using 5 criteria in our

RE (see below), we wanted to adopt a viable yet proportional representation of these criteria when it came to their respective contributions into the list of suggestions. By and large, this limit for k may be highly dependent on the choices made by the portal and so, it can be reconfigurable.

The 5 distinct and likely weighted categories `PolyRecs` depends on to create its recommendations are the following:

1. A visitor's favorite categories (weight w_1),
2. Articles whose categories appear to be "strongly related" to the current category being read (w_2)
3. The time of day a visitor browses a specific piece of content on the portal (w_3),
4. Articles that share a high *cosine*–similarity value to the one currently in access (w_4), and finally,
5. Articles that are popular today (w_5).

`PolyRecs` stores aggregated data associating each visitor with the number of times she has accessed a specific category. Using such counters, pieces of content are fetched that are tagged with categories present on the top 85% of her most frequent reads. The above percentage was set after observing that the lowest 15% of the categories a visitor views, usually have very low counters indicating negligible impact.

A category's strong or weak relation to one another, is generated based on the number of times one has been read after the other. Tracking each visitor's behavior throughout `athensvoice.gr`, `PolyRecs` is able to create this correlation between categories of the portal and return pieces of content from pertinent categories. Each visitor's click in the portal is also associated with the time at which it occurred. This allows `PolyRecs` to associate categories frequently accessed at specific times of day (e.g., articles that describe recipes, or restaurants are more frequently read during midday). In `PolyRecs` *DBMS* data concerning the popularity of each piece of content is stored. This allows the *RE* to suggest articles that show a high popularity today. The criterion utilizing the similarity between articles is based on the *cosine* similarity value. Having this value available for each piece of content present on `athensvoice.gr`, `PolyRecs` can recommend articles that share a high value to the one a visitor is currently reading.

In the above 5 categories, we should have that $\Sigma(w_i) = k$ holds true at all times. Although we could use an equal-weighted approach for these w_i, in the context of `athensvoice.gr`, we empirically found that a slightly different weight vector does work more effectively in our applications setting; this vector is $w =< 4, 3, 3, 3, 2 >$.

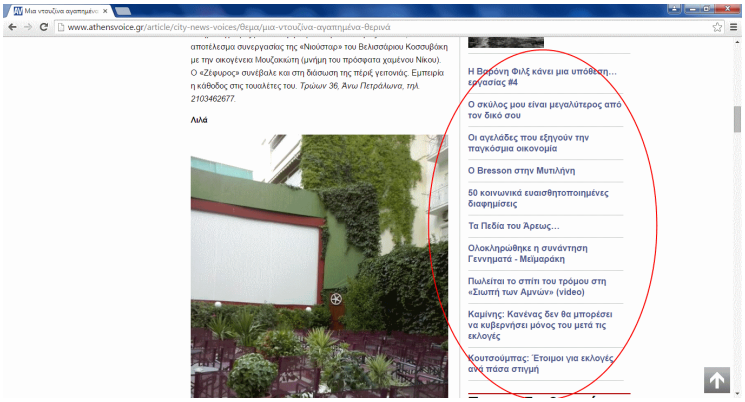
We should also point out that content that has been already browsed by the user at any point in her dwelling on the portal is explicitly excluded from the suggestions. In the final compilation of recommendation, we also do carry out duplicate elimination and if needed our engine brings in additional results to appropriately fill in the list of suggestions. As items are continually introduced to the portal through the *Drupal* CMS, their meta-data pass on to the `PolyRecs` database and so they immediately become available for the computations performed by *RE*.

4.3 `PolyRecs` Interface Handler (*IH*)

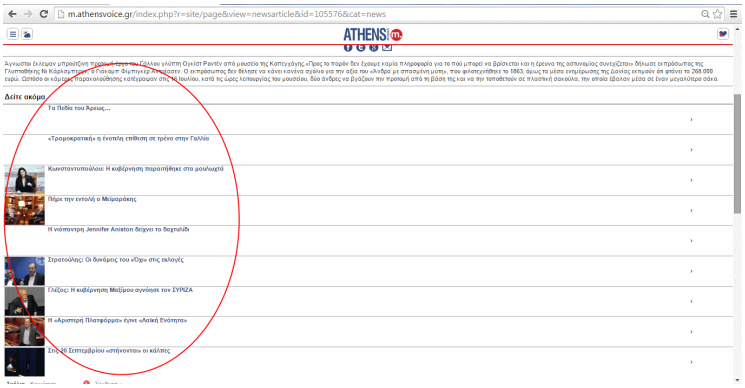
IH is responsible for the presentation of the recommendations coming from the core of `PolyRecs` and helps with the rendering of this result set on specific devices. Initially, the interface handler receives the triplet of data generated during the browsing of an article and passes this information as input to `PolyRecs`. With its turn, the engine sends out

the recommendation list for the article in question and the *IH* manages the generation of the required `HTML` snippet; the snippet also contains style-sheet rules as well as automatically generated *JavaScript* event-functions. The snippet is then transported to the consuming *Apache* web-server for final dispatching to the user device (Figure 1).

Figure 2 depicts how the result-sets for recommendations appear to either a desktop or mobile device. Evidently, when the outcome of a presentation may have multiple forms which are generated with the intervention of the *IH*. The specifics of the rendering involved here are exclusively an assignment taken over by the interface handler. In addition to the suggestions, each specific recommendation on the user-device is escorted by a specific *JavaScript* event function whose goal is to inform the `PolyRecs` database if and where a user elects to click and go further. This *JavaScript* mechanism provides the capability for `PolyRecs` to very accurately account for clicks induced from the lists of provided recommendations.



(a) Recommendations Area for Desktop/Tablet Visitors



(b) Recommendations Area for Mobile Device Visitors

Fig. 2: Recommendation Areas for Various Devices

4.4 PolyRecs Maintenance System (*Msys*)

The maintenance component consists of a number of `php`–scripts that get executed within `PolyRecs` and are mostly *cron*-jobs. Among those tasks, the most important are:

• *User Profile Reset* cleanses a user's profile from data that are at some point considered noise. These data are about category reads that rank significantly low to make it into the output. To this end, they have essentially ceased to characterize a visitor's active preferences.

• *Content Similarity Generation* performs the respective computations among pieces of content. As this is a computationally intensive task [8], we perform it as a *cron*-job every 20 minutes; over long periods of time we have ascertained that every 20 minutes a new article appears on the average on `athensvoice.gr`.

• *Plasticity Solver* aims at addressing the problem of rigid preferences attributed over time to a user. The problem appears in content–based or collaborative filtering recommendation algorithms and occurs when after some time, the preferences of a user are practically impossible to change despite the fact that her unique visits to article has been documented. To prevent this phenomenon, *MSys* runs this solver to reduce the counters stored in cumulative tables by 25% for those values that display a big difference (e.g., >80%) with other rows in the respective tables. Typically, the solver is run every few days.

• *Delayed Updater* gets to execute updates needed mostly for non-critical data and produces aggregates of information at the end of the business day. Over time, we observed that during peak-periods, `PolyRecs` used to produce recommendations at a slower pace. To remedy this, we opted to stage less critical data update operations to a batch file that was ultimately executed once a day in off-peak hours. For example, there is no urgency to increment values of statistics for category correlation that are already high; this operation receives delayed treatment by being relegated to the *Delayed Updater*.

4.5 PolyRecs Injector (*Inj*)

This component furnishes a mechanism that is very essential in the evaluation of `PolyRecs` effectiveness. The *RE* module produces its recommendations based on the 5 earlier-stated criteria. It is however crucial that a feedback mechanism be established so that `PolyRecs` ultimately becomes aware of which suggestions as well as the corresponding criteria were ultimately used out of the recommendation list. In this respect, *Inj* maintains statistics in terms of counters and instruments the list of recommendations bound for the *IH* by automatically adding *JavaScript* code for every item on this list. A *JavaScript* event function is generated on-the-fly and gets attached to every recommendation produced.

Figure 3 shows the result of *Inj*'s instrumentation of the recommendations; this outcome will finally make it through the *IH* and *Apache* web-server to the user's browser. As Figure 3 indicates, this function takes as input 3 parameters: *1)* the *visitor-id* for whom the recommendations were created, *2)* the *id* of the article being recommended, and *3)* the *criterion-id* (i.e., numbers 1..5) by which the suggestion was generated. If a browsing user clicks a recommendation, the accompanied *JavaScript* code-snippet is executed. As a consequence, the aforementioned 3 pieces of information along with the time-stamp of the click are inserted into `PolyRecs`'s *DBMS* to be evaluated at a

later stage. The evaluation is performed to measure the performance of each criterion integrated into PolyRecs.

```
<a href="/article/design-home/article/open-house-athens-2015"
onclick="clicked(123,109891,2)">Open House Athens 2015</a>
```

Fig. 3: PolyRecs *Injector* Snippet Addition

We have found that this method of providing instrumentation for evaluation does offer two clear advantages:

1. By tracking the clicks of each visitor, we can more effectively evaluate the PolyRecs performance and analyze the possible benefits in terms of *CTR* (click-through-rate).
2. By keeping track of the criterion by which every click was produced, we can better quantify the effect each criterion has on PolyRecs.

In addition, this approach does offer opportunities for extensions by for instance being able to integrate seamlessly in the future more sophisticated techniques (such as machine-learning) to help automatically compute the weights for each criterion over time.

5 Evaluation

In evaluating the PolyRecs prototype, we aim at establishing the effectiveness of our overall approach, in pointing out the benefits of our hybrid proposal and in assessing PolyRecs's contribution to the overall running of the athensvoice.gr portal. Using Google's Analytics and having access to both PolyRecs-enabled and stand-alone versions of the production portal, we are able to present page-view rates over lengthy periods of observation. Google's Analytics is an independent service commonly used by web-based systems to gain insight on operational trade-offs and evaluate performance. In this section, we also gauge the merit of integrating PolyRecs into the production portal through a succinct cost/benefit analysis.

5.1 Analysis with Google's Analytics

We commence by examining the number of page-views the portal received over a 1-month period while using the PolyRecs engine and compare these page-views with their counterparts that occurred a year before. In this, we investigate the traffic generated by both desktop/tablet and mobile devices. We are interested in this grouping for two reasons: firstly, the web-page lay out is different for these two types of machinery and secondly, user-behavior has been shown to differ when hand-held devices are used for browsing [20].

The reported time periods are those of *24/07/2015-24/08/2015* and *24/07/2014-24/08/2014*. We selected this specific day-span so as to remain as much as possible unaffected by user behavior; by and large, this period coincides with the summer vacations in the country.

Table 1 depicts the increase observed in page-views per session for the two classes of devices used for access. We should note here that we have managed to augment the rates despite a 9.58% experienced drop in the unique number of visitors over the 2015 period. On average, `PolyRecs` manages to deliver an increase of 45.81% in combined page-views per session.

Table 1: Increasing Page-Views/Session through `PolyRecs`(Month Period)

Page-Views/Session with `PolyRecs` enabled	Desktops & Tablets	Mobile Devices
Page-Views/Session Increase	+73.81%	+17.81%

Figure 4 shows how the page-views/sessions fared for each day of the observation period when only desktop and tablet traffic was taken into consideration. Clearly, there exists an indisputable positive effect from the deployment of `PolyRecs` as the curve from the 2015 remains consistently higher at all times from that of 2014. Similar results

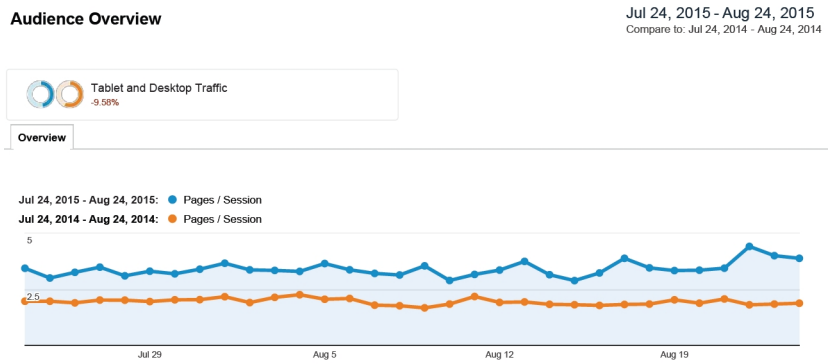


Fig. 4: Chart for Average Daily Page-View/Session for Desktop/Tablet Users

can be seen in Figure 5 that depicts the respective curves when we analyze activity generated only by mobile devices. The `PolyRecs`-enabled curve invariably produces improved page-views per session rates throughout the month.

In an effort called "4-day experiment", we seek to ascertain the value of `PolyRecs` within a short and recent period of time. Hence, we collected operational statistics for two days (9/22-9/23) while having `PolyRecs` activated and then, we repeated the same exercise with the engine not in operation. on the same days of the following week ($9/29 - 9/30$). Table 2 shows an aggregation of the results obtained. We establish an increment in page-views/session rates for both desktop/tablet and mobile classes when `PolyRecs` is enabled although the gain in the mobile devices is limited. Overall, `PolyRecs` demonstrates its value by offering a higher by 6.3% average page-views per session. Figure 6 depicts an overview of the change in terms of the page-views/session metric for all portal users in the above 4-day experiment. The metric does remain

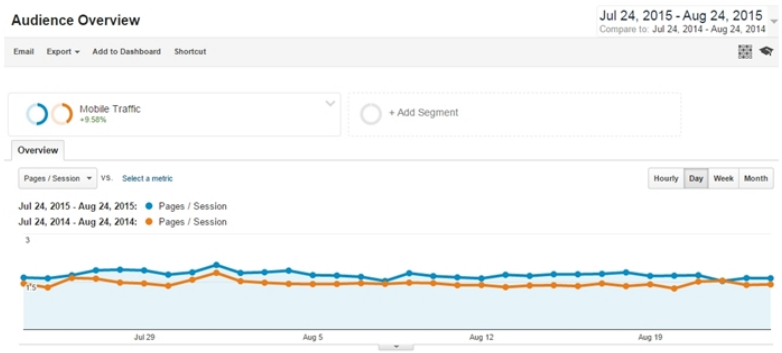


Fig. 5: Chart for Average Daily Page-Views/Session for Mobile Devices Visitors

Table 2: Page-Views/Session and Unique Visitors

	Desktop/Tablet without PolyRecs 9/29-9/30	Desktop/Tablet with PolyRecs enabled 9/22-9/23	Mobile Devices without PolyRecs 9/29-9/30	Mobile Devices with PolyRecs enabled 9/22-9/23
Page-Views/Sess.	2.83	3.18	1.42	1.46
Unique Visitors	167,240	148,921	117,003	128,784

higher during the period that the engine is on. It is also worth pointing out that during the period of 9/22-9/23 all rates are higher for both user groups than those attained in 9/29-9/30.

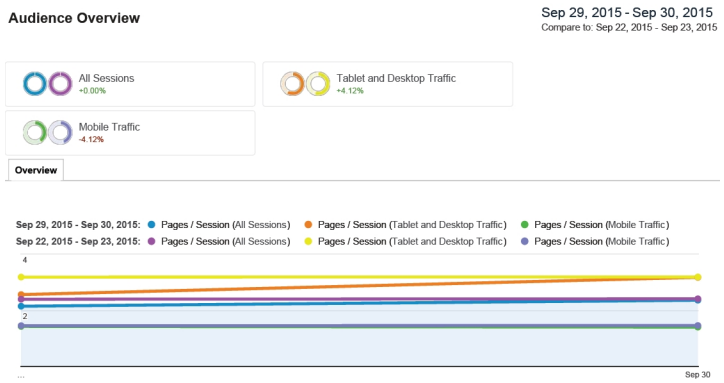


Fig. 6: Google's Analytics-chart Comparing Page-Views/Session for the 4-day Exper.

Table 3 outlines the "deactivation effect" in terms of percentages for both page-view/session as well as average dwell time on the portal (delivered by Google's Analytics). By in-activating the recommendation engine, the page-view/session rates do fall but more importantly, visitors spent less time reading content from the portal.

Table 3: `PolyRecs` Deactivation Effect

	Desktop/Tablet Class	Mobile Devices Class
Page-Views/Session	-10.95%	-2.85%
Average Time Spent	-17.76%	-10.49%

5.2 Assessing the Impact of the Recommendation Criteria

In this section, we analyze the impact the 5 criteria used by the engine in the compilation of recommendation lists. For this endeavor, we use data harnessed by the portal in a 12-day period of 10/10/2015–10/22/2015. Through the use of *JavaScript* functions, we were able to collect accurate statistics on how impactful the 5 criteria are.

Figure 7 shows how each of the criteria used (x-axis) fared in terms of clicks (y-axis) given that the two classes of user-devices were used for accessing. As anticipated, numbers of recommendations served by each criteria do differ. During the above period of monitoring, a total of 74,030 number of clicks were logged. From those approximately 40,000 were originated by mobile devices and the rest were coming from both desktop and tablet users.

In general, the first two criteria (1 and 2) contribute a great deal in the suggestions offered. Much help also comes from criterion 5 despite the fact that is assigned a smaller weight (w_5=2). Desktop and tablet visitors show a strong preference for the

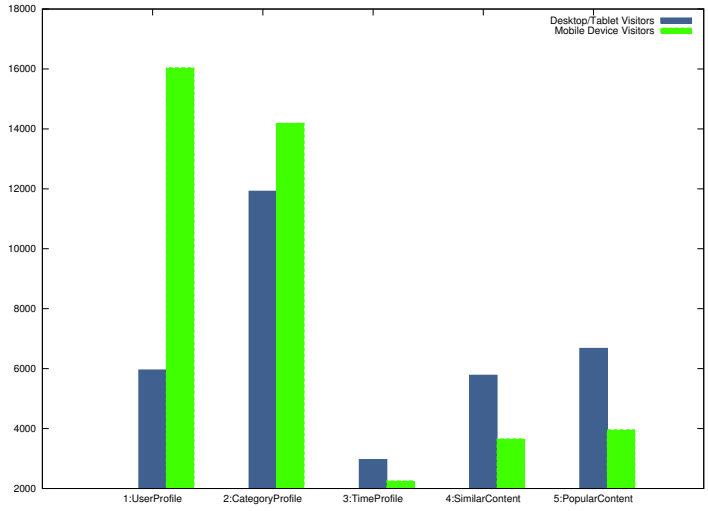


Fig. 7: `PolyRecs` Criteria Impact

criterion that profiles the categories of articles (2). The number of clicks produced by this criterion, is almost double of the next best which is that of the most popular content (5). For the mobile device group of visitors, the user profile criterion (1) ranks first in for the number of clicks generated; also, criterion 2 comes a close second whereas

the remaining three come in close with each other with an average of approximately 3,300 clicks a piece. It is worth noting that in for the "user profile" category (i.e., 1), there is an impressive difference between the clicks of mobile and desktop users; while desktop/tables contribute 5,956 clicks, the mobile devices generate 16,040 clicks.

The pie-chart of Figure 8 depicts how the criteria used in the engine fared regardless of the class of user devices. Approximately 35% of the user clicks to `PolyRecs` recommendations were attributed to the "category criterion" (e.g., 2) with another 30% coming off the "user profile" criterion (1); all other three criteria scored below 15%.

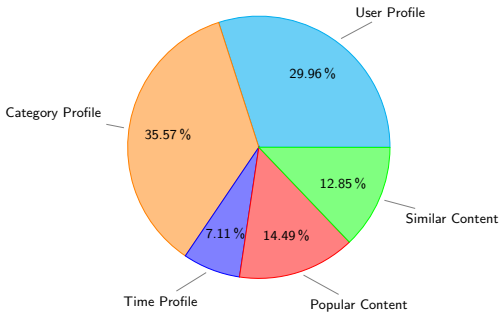


Fig. 8: Percentages of User Clicks due to 5 Recommendation Criteria

5.3 Benefits of Integration

To establish the financial benefits of having `PolyRecs` incorporated in `athensvoice.gr`, we provide a cost/benefit analysis based on two scenarios: the best in which all clicks made out of recommendations furnished by `PolyRecs` and a worst scenario in which only 25% of clicks in discussion finally occurred.

We should also approximately calculate the number of impressions (i.e., page-views) produced for a month period using the aforementioned 12-day period of monitoring. In this, `PolyRecs` logged 74,000 impressions. By extrapolation, we derive a total of 185,000 new page views. `athensvoice.gr` features 6 slots for for advertisement banners into every page (1 skin, 1 728x90 banner, 3 300x250 banners, 1 text link). These slots, if sold by the advertisement department, have an average *CPM* (cost per million impressions) of 1.50€; every slot's cost is determined by the position it has on the webpage. Figure 9 shows the benefit reaped by integrating `PolyRecs` into the portal. On a yearly basis, the respective values become 4,994€ in the worst case and 19,980€ in the best case–a sizable benefit in either case. We should also take into account the expenses needed for `PolyRecs`'s hosting, which is at 69€ on a monthly basis; approximately the revenue from one advertising banner in the worst case scenario.

The above projected benefits are highly dependent of the following factors: *a)* should `PolyRecs` be integrated with a portal enjoying a higher traffic, it would bring in higher income, *b)* the limited number of recommendations served by the *RE* and their style/-format, *c)* the placement of `PolyRecs`'s recommendation on the page rendered. Without doubt, the slot within which the suggestion appears, plays a key role in the viewing

Revenue Per Month	Worst Case (46,250 impr.)	Best Case (185,000 impr.)
1 banner	69.38€	277.50€
6 banners	416.30€	1,665.00€

Fig. 9: Economic Benefit derived in Best/Worst Case Scenarios

and the likely clicking by the users, and *d)* the total number of advertisement slots on web page and the average expected *CPM*.

6 Conclusions and Future Work

In this paper, we present `PolyRecs`, a hybrid recommendation system that deploys multiple criteria to produce suggestions for a popular news, politics, arts, opinion articles and discussion portal in Greece. We have placed particular emphasis on the requirement that recommendations have to be delivered within very strict time constraints in order to realize a viable approach for improving pages-views and traffic for the `athensvoice.gr` publication. We have developed `PolyRecs` in a way that it can be successfully integrated with any contemporary data infrastructure consists of modern `CMS` and database systems. We have deployed the prototype and have used the resulting production system to collect statistics and ascertain the utility of our proposal. While observing the behavior of the prototype, we have established that the use of `PolyRecs` in the *AthensVoice* portal has led to average in increase of 6.3% in the page-views consumed by visitors. In terms of numbers, there was a total of almost $74,000$ clicks increase in a period of 12 days.

In the future, we would like to extend `PolyRecs` in a number of ways. More specifically, we plan to: *a)* embed social media features and so become able to access demographic characteristics for visitors; this can clearly offer a wealth of personalized information for creating more effective suggestions, *b)* exploit user location and use geographically-pertinent articles to furnish more focused recommendation criteria, *c)* experiment with spots for listing recommendations to further enhance *CTR*-rates, and *d)* use machine learning approaches on the accumulated statistics to drive the weighting scheme in a more sophisticated and likely more effective way.

Acknowledgements: we are grateful for the reviewer comments received; partial support for this work has been provided by the *GALENA EU* Project and *Google*.

References

1. Richard McCreadie, Craig Macdonald, and Iadh Ounis. News vertical search: When and What to Display to Users. In *Proc. of 36th Int. ACM SIGIR Conf.*, Dublin, Ireland, 2013.
2. Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. In *Proc. of 29th Int. ACM SIGIR Conf.*, Seattle, WA, August 2006.
3. Jiahui Liu, Peter Dolan, and Elin Ronby Pedersen. Personalized News Recommendation Based on Click Behavior. In *Proc. of 15th Int. Conf. on Intelligent User Interfaces (IUI)*, Hong Kong, PR China, 2010.

4. Andrii Maksai, Florent Garcin, and Boi Faltings. Predicting Online Performance of News Recommender Systems Through Richer Evaluation Metrics. In *Proc. of 9th ACM RecSys Conf.*, New York, NY, 2015.
5. Zhongqi Lu, Zhicheng Dou, Jianxun Lian, Xing Xie, and Qiang Yang. Content-Based Collaborative Filtering for News Topic Recommendation. In *AAAI Conf.*, February 2015.
6. Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan-Nan Liu, and Suju Rajan. Beyond Clicks: Dwell Time for Personalization. In *Proc. of 8th ACM RecSys Conf.*, Foster City, CA, 2014.
7. Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. Inferring and Obfuscating User Gender Based on Ratings. In *Proc. of 6th ACM RecSys Conf.*, Barcelona, Spain, September 2012.
8. Karsten Schmidt, Sebastian Bächle, Philipp Scholl, and Georg Nold. Big Scale Text Analytics and Smart Content Navigation. In *Proc. of the 2013 BIRTE Workshop*, pages 167–170, Riva del Garda, Italy, September 2013.
9. Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google News Personalization: Scalable Online Collaborative Filtering. In *Proc. of 16th Int. Conf. on WWW*, Banff, Canada, 2007.
10. Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. Offline and Online Evaluation of News Recommender Systems at `swissinfo.ch`. In *Proc. of 8th ACM Conf. on RecSys*, Foster City, CA, 2014.
11. Weinan Zhang, Jun Wang, Bowei Chen, and Xiaoxue Zhao. To Personalize or Not: a Risk Management Perspective. In *Proc. of 7th ACM RecSys Conf.*, Hong Kong, China, 2013.
12. Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. Personalized Next-song Recommendation in Online Karaokes. In *Proc. of 7th ACM RecSys Conf.*, Hong Kong, PR China, October 2013.
13. Saul Vargas and Pablo Castells. Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. In *Proc. of 5th ACM RecSys Conf.*, Chicago, IL, 2011.
14. Florent Garcin, Christos Dimitrakakis, and Boi Faltings. Personalized News Recommendation with Context Trees. In *Proc. of 7th ACM RecSys Conf*, Hong Kong, China, 2013.
15. Fernando Diaz. Integration of News Content into Web Results. In *Proc. of 2nd ACM Int. Conf. on WSDM*, pages 182–191, Barcelona, Spain, 2009.
16. Arnd Christian König, Michael Gamon, and Qiang Wu. Click-through Prediction for News Queries. In *Proc. of the 32nd Int. ACM SIGIR Conf.*, pages 347–354, Boston, MA, 2009.
17. Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean-Francois Crespo. Sources of Evidence for Vertical Selection. In *Proc. of 32nd Int. ACM SIGIR Conf.*, pages 315–322, Boston, MA, 2009.
18. Alan Said, Alejandro Bellogin, Jimmy Lin, and Arjen de Vries. Do Recommendations Matter?: News Recommendation in Real Life. In *Comp. of 17th ACM Conf. on CSCW on Social Computing*, pages 237–240, Baltimore, MD, 2014.
19. G. Gebremeskel and A. P. de Vries. The Degree of Randomness in a Live Recommender Systems Evaluation. In *Working Notes, Conf. and Labs of the Evaluation Forum (CLEF)*, Toulouse, France, September 2015.
20. Yang Song, Hao Ma, Hongning Wang, and Kuansan Wang. Exploring and Exploiting User Search Behavior on Mobile and Tablet Devices to Improve Search Relevance. In *Proc. of 22nd Int. WWW Conf.*, pages 1201–1212, Rio de Janeiro, Brazil, 2013. ACM.