NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

THESIS

A Hybrid Approach to Recommendation Systems using Multiple Criteria

Michail Papakonstantinou

Supervisor: **Alexis Delis,** Professor UoA

ATHENS

FEBRUARY 2016

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υβριδική προσέγγιση στα συστήματα συστάσεων με χρήση πολλαπλών κριτηρίων

Μιχαήλ Παπακωνσταντίνου

ΕΠΙΒΛΕΠΩΝ: **Αλέξης Δελής,** Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΦΕΒΡΟΥΑΡΙΟΣ 2016

THESIS

A Hybrid Approach to Recommendation Systems using Multiple Criteria

Michail Papakonstantinou
ID: M1315

SUPERVISOR: **Alexis Delis,** Professor UoA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υβριδική προσέγγιση στα συστήματα συστάσεων με χρήση πολλαπλών κριτηρίων

Μιχαήλ Παπακωνσταντίνου
Α.Μ.: Μ1315

ΕΠΙΒΛΕΠΩΝ: **Αλέξης Δελής,** Καθηγητής ΕΚΠΑ

ABSTRACT

In this work, we describe the development of a hybrid recommendation system that uses multiple criteria for its recommendations. The system utilizes a list of 5 criteria and is focused on being able to work real-time, for the recommendations produced. It combines both online and offline features as part if its work is done real-time and part of it offline through cron jobs. We deploy this system on the website of athensvoice.gr, in order to assess its effects on a real website scenario, explore the correlation between different categories of the posts on the website and analyse the results it offers. To this end, we propose an approach to ascertain the usefulness of our results as they are currently produced by our production website. We consider the work described in this thesis novel since it is developed, run and evaluated on a production website, covering a variety of articles. Furthermore, through the multiple criteria used, our system shows an improved behaviour on problems of traditional methods.

SUBJECT AREA: Recommendation Systems

KEYWORDS: hybrid recommendation systems, real world use case, multiple criteria algorithm, content based collaborative, real-time evaluation

ΠΕΡΙΛΗΨΗ

Στην εργασία αυτή περιγράφουμε την ανάπτυξη ενός υβριδικού συστήματος συστάσεων που παράγει προτάσεις με χρήση πολλαπλών κριτηρίων. Το σύστημα χρησιμοποιεί ένα σύνολο ποιοτικών κριτηρίων και είναι εστιασμένο στο να μπορεί να παράγει τις συστάσεις του σε πραγματικό χρόνο. Συνδυάζει χαρακτηριστικά πραγματικού χρόνου και παρασκηνίου, καθώς μέρος των εργασιών του εκτελείται κατά την παραγωγή των συστάσεων αλλά και μέρος στο παρασκήνιο. Για να μπορέσουμε να αξιολογήσουμε τις επιδράσεις λειτουργίας του συστήματος, το θέσαμε σε χρήση στον ιστότοπο της Athens Voice, ώστε να μπορέσουμε να αναλύσουμε τη σχέση των κατηγοριών που προσφέρονται, των άρθρων που βρίσκονται εκεί αλλά και να ποσοτικοποίησουμε τα αποτελέσματα εφαρμογής του. Για το σκοπό αυτό προτείνουμε μια μέθοδο για την επαλήθευση της αξίας των παραγόμενων αποτελεσμάτων από την εφαρμογή του συστήματος σε ένα υπάρχον ιστότοπο. Θεωρούμε καινοτόμα τη δουλειά που περιγράφεται στην εργασία αυτή, καθώς αναπτύχθηκε, εφαρμόστηκε και εξετάστηκε κάτω από πραγματικές συνθήκες, σε ιστότοπο που καλύπτει μια μεγάλη ποικιλία περιεχομένου. Επιπλέον μέσω των πολλαπλών κριτηρίων που χρησιμοποιεί, το σύστημα συστάσεων που αναπτύχθηκε παρουσιάζει βελτιωμένη συμπεριφορά σε προβλήματα των παραδοσιακότερων τεχνικών.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Συστήματα Συστάσεων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: υβριδικά συστήματα συστάσεων, περίπτωση χρήσης πραγματικού κόσμου, αλγόριθμος πολλαπλών κριτηρίων, με βάση το περιεχόμενο συνεργατικό, αξιολόγηση πραγματικού χρόνου

ACKNOWLEDGMENTS

I would like to thank supervisor, Alexis Delis for his continuous cooperation and help throughout the preparation and completion of this thesis. I would also like to thank my colleagues at Athens Voice and especially Vaios for their interest in deploying the current work into the website.

CONTENTS

1. INTRODUCTION 13

2. THEORETICAL OVERVIEW 15

2.1 Content Based Systems 15

2.2 Collaborative Filtering 16

2.3 Hybrid Filtering 17

3. RELATED WORK 18

4. HIGH LEVEL DESCRIPTION OF RECOMMENDATION SYSTEM 21

5. TECHNICAL OVERVIEW OF RECOMMENDATION SYSTEM 23

5.1 Drupal System 24

5.2 Interface Subsystem 24

5.3 Recommendation System Database Overview 26

5.3.1 Tracker Schema 26

5.3.2 Backup Schema 29

5.4 Recommendation Engine 31

5.5 Backend Engine 32

5.6 Custom Metric Description 33

6. RECOMMENDATION ALGORITHM 35

6.1 System Data Overview 35

6.2 Data Usage 36

6.3 Data Evaluation, Final Recommendations 37

6.4 System Complexity Overview 38

6.4.1 initdata.php 39

6.4.2 render.php 40

6.4.3 run_queries.php 41

7. RESULT ANALYSIS 43

7.1 General Statistics Overview 44

7.2 Analysis using Google Analytics 44
7.2.1 System in Use Analysis 45
7.2.2 System Deactivation Analysis 47

7.3 Custom Analysis 49

7.4 Benefits of Integration 53

8. CONCLUSION AND FUTURE WORK **55**

8.1 Conclusion 55

8.2 Future Work 55

Appendix A. TRACKER SCHEMA TABLE DESCRIPTION **56**

Appendix B. CLICK TABLE DESCRIPTION **62**

REFERENCES **63**

LIST OF FIGURES

Figure 1: High Level System Diagram 21

Figure 2: Algorithmic Overview of RecSys 22

Figure 3: Recommendation System Overview. 23

Figure 4: Recommendation’s area for desktop/tablet visitors. 25

Figure 5: Recommendation’s area for mobile device visitors. 25

Figure 6: Complete database. 27

Figure 7: Example of onclick JavaScript function 34

Figure 8: Algorithmic Overview of Data Handling 36

Figure 9: Algorithmic Overview for Recommendations Produced 39

Figure 10: Execution time in milliseconds for initdata.php 40

Figure 11: Execution time in milliseconds for render.php 41

Figure 12: Data Statistics of the System 45

Figure 13: Chart for Desktop/Tablet visitors 46

Figure 14: Percentage Increase for Pages/Session (Desktop/Tablet visitors) . 46

Figure 15: Chart for Mobile Devices visitors 47

Figure 16: Percentage Increase for Pages/Session (Mobile Devices visitors) . 47

Figure 17: Comparison chart deactivated/activated System 48

Figure 18: Percentage Increase for Pages/Session with System activated . . . 48

Figure 19: Average time spent on website during deactivation of system 49

Figure 20: Click criteria choice for desktop/tablet users 50

Figure 21: Click criteria choice for mobile device users 50

Figure 22: Percentages of click criteria by athensvoice.gr desktop/tablet users 51

Figure 23: Percentages of click criteria by athensvoice.gr mobile devices users 51

Figure 24: Accumulated percentages of click criteria by athensvoice.gr visitors 52

Figure 25: Clicks on recommendations based on time of day 53
Figure 26: Economic benefit of integration in best/worst case scenario 53
Figure 27: Table initdata. 56
Figure 28: Table user_from_to. 56
Figure 29: Table user_has_read. 57
Figure 30: Table user_from_nid_to_nid. 57
Figure 31: Table user_profile. 57
Figure 32: Table time_tid. 58
Figure 33: Table time_profile. 58
Figure 34: Table tid_profile. 59
Figure 35: Table nid_profile. 59
Figure 36: Table cosine_sim. 59
Figure 37: Table day_tid. 60
Figure 38: Table day_profile. 60
Figure 39: Table node_counter. 60
Figure 40: Table click. 62

1. INTRODUCTION

Content recommendation systems have recently played a key role in the areas of news and report dissemination as they better serve the individual needs of their user communities. Being proactive and furnishing a user with pertinent content she might be interested in a timely fashion does benefit the content-provider as user spends longer time on the website, revenues from advertisements are likely to increase and CTR (click-through-rate) will follow suit. As a result the interest in recommendation systems has increased over the years.

In this thesis, we develop such a recommendation system and deploy it on the website of athensvoice.gr. Athens Voice started as a free weekly newspaper distributed in various areas of Athens, Greece, and over the years became a website covering both news and a variety of topics and features. In this recommendation system, we are using both content-based and collaborative filtering, yielding a truly hybrid system. 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 a live website.

Until now, the articles suggested towards the visitors of the website are served using popular criteria for such websites, such as:

- based on the time the articles was posted
- fetching more articles from the same category
- human chosen articles as featured
- more from the same author, etc.

Since none of these criteria takes into account the uniqueness of each visitor, we expect the use of our recommendation system to have a positive effect towards the athensvoice.gr visitors, and an increase of the number of pages per visitor to be seen as a result.

The developed system has both online/real-time and offline parts. It serves the recommendations produced online, as the visitor scrolls within a web-page, at a pre-specified part of the page, which differs across devices. Keeping the system's responsiveness in mind, we divided the computational work in two parts, the online and the offline one. The work performed online has to do with the significant factors that need to be taken into account real-time, i.e. excluding from the results the articles already read by a specific user. The offline work, which is done through cron jobs, involves less time crucial computation, such as increasing the correlation between categories which is already high.

Our work differs from what has been proposed so far in a number of aspects. The system developed is not solely focused on news recommendations, although news posts are part of the system, these posts are not the only ones handled by it. Another important factor that makes this work novel, is that due to the nature of the recommendation algorithm used (as explained later, a multiple criteria one), the system was in use from the first day of its deployment on the website of athensvoice.gr, both collecting data (and improving its recommendations as a result) as well as serving its suggestions towards each visitor of the website. Moreover due to the variety of criteria it uses, the handling of traditional problems of recommendation systems is done in a more efficient way. This work, even though carried out independently, is deployed and run on a real website, with real-time visitors. This is important to keep in mind, since its training, evaluation and serving of recommendations is done real-time on a live website. Another important contribution of this work, is its novel metric for the assessment of recommendations. The metric developed uses the visitor's click as a main factor, and it tracks the criteria by which the recommendation clicked was produced, giving the system a self-evaluation method.

The structure of this thesis is as follows:

- Theoretical overview of recommendation systems.
- Comparison with related work and differences in approach.
- High level description of the recommendation system, to explain the flow and storage of data.
- Technical description of the system developed.
- Description of the recommendation algorithm implemented.
- Analysis of the results.
- Conclusion and future work.

2. THEORETICAL OVERVIEW

An increasing interest over the years has been observed for systems that can predict a user's preferences, towards the content of a web site. Such systems are called recommendation systems, and their goal is to accurately suggest content to users that may possibly interest them.

Recommendation systems may be applied to a variety of fields, and serve their suggestions regardless of the content they are suggesting. Common applications of such systems are:

- Product recommendation, a very common application of recommendation systems are e-shops that recommend products based on the user's preferences and by showing items that are frequently bought together (for example ebay and amazon use such systems).
- Movie recommendation, another popular application of recommendation systems is companies that suggest movies that their user's might want to see. Netflix is such an example, whose recommendations are based on user ratings, in order to be calculated and served.
- News recommendation, another application of these systems is recommending news articles. This field has received increasing interest in the past years, since companies like Google or Yahoo!, have adopted recommendation systems in order to present their users with articles that might interest them.

Recommendation systems may use a number of techniques for their suggestions to be served. Based on these techniques, they can be classified in the following 3 general classes:

- Content-based systems, which are based on the properties of a specific piece of content.
- Collaborative filtering systems, which are based on the similarity between different content and/or users.
- Hybrid approaches, that combine features of the above techniques.

2.1 Content Based Systems

Content based recommendation systems, depend on the properties and characteristics of each piece of content in order to make their recommendations towards the users. For that to be possible, the content profile has to be constructed, that contains these characteristics [8, 3].

These characteristics may vary based on the area in which the recommendation system is deployed. For example, when creating a content based system for movies, the characteristics of each content (movie), may contain:

- Genre.
- Actors.
- Director, etc.

Another example of the usage of such systems is when building a user profile, based on his/hers preferences. Again there are a number of characteristics that can be used, such as:

- User's ratings.
- Past buys.
- Previous views, etc.

Having these profiles, a recommendation system can produce its suggestions by calculating the degree in which a user may be interested in a specific piece of content. This can be done in various ways and algorithms, like: averaging values of rated items, Bayesian Classifiers, Cluster Analysis, Decision Trees, Neural Networks, etc.

The use of such recommendation systems offer a number of advantages:

- Taking into account each user's preferences, the suggestions made are accurately matched with the interests of each user of the system.
- Performing heavy calculation for these systems offline, leads to an increase in real-time performance.
- Recommending new pieces of content to users, since these systems use the specific item's characteristics and do not require them to be heavily rated by other users .

On the other side, the use of content based recommendation systems, have some disadvantages:

1. The cold start problem, which happens when a new user enters the system, for whom no data is available and as a result no accurate suggestion can be made.
2. In contexts where characteristics of the content is not available or is scarce (e.g. images, sounds), these systems are unable to recommend content with accuracy since no properties are present to make the recommendation.
3. These systems are also subject to the stability vs plasticity problem[14], which is when a user has rated a large number of items, his/hers preferences into the system are difficult to change, whereas in real-time cases this can happen pretty often

2.2 Collaborative Filtering

Collaborative filtering systems, are a different approach for recommendation systems than content based ones. They differ in the sense of the input they use in order to make their suggestions. Such systems focus on the similarity between users or items and recommend content based on that [3, 8, 2].

These systems use a distance measure such as Jaccard or cosine distance in order to calculate the degree of similarity between users and items. This way, collaborative approaches are unaffected by the context in which they are deployed, since they do not take into account the specific properties of the content they are recommending (like content-based systems), but only the way users interact with it. They also do not suffer by the cold start problem for new users, since they can be matched with existent ones for whom a lot of data is present and make the recommendations based on it. Finally they handle contexts in which the properties of the content suggested is scarce with high accuracy since they do not need to take into account the specific content's properties to make their recommendation.

However the use of these systems has some disadvantages:

1. The cold start problem (for a new item), which is when a new item is introduced into the system and for which there is no data available in order to include it in the computation of the recommendation.
2. The data stored into the tables tend to be sparse. This happens if the number of users is not proportional to the number of items. In order to counter this problem, special techniques are being deployed.
3. The grey sheep problem [7], which is when a user with unusual preferences enters the system, who is difficult to match with the existent ones.

2.3 Hybrid Filtering

Hybrid filtering systems combine one or more techniques and properties of both content-based and collaborative filtering systems. This is done in order to overcome most if not all of the disadvantages of each system separately [3, 2].

For example, these systems can overcome the cold start problem for new users since for them a collaborative approach can be used to make the recommendations as accurate as possible. The same way the cold start problem for new items can also be reduced if the item's properties are taken into account. Like collaborative filtering systems, hybrid ones, do not necessarily require the presence of the content's properties since they can use a collaborative approach to make recommendations in such contexts.

In this work, we develop a hybrid recommendation system that uses a number of criteria to make its recommendations. Through these 5 criteria, we aim to overcome the common disadvantages of other recommendation systems, while trying to keep the speed of this system as high as possible, since it is deployed on a live website with real-time visitors.

3. RELATED WORK

Most of the work done in this field has focused on news recommendations [11, 12, 6, 10, 24, 9, 1, 5], as well as possible metrics [27, 23, 20, 4, 11, 10, 25] to evaluate the effect of the proposed algorithm. The recommendation system presented in this work, handles news articles as a special case of articles due to their specific nature and their rather short life span. Our system is applied throughout the website of athensvoice.gr, providing recommendations for the content published on the website. This is a main difference between the current and related work being done in this field.

As stated in the introduction, our system is run real-time, evaluating and serving recommendations on the fly to users during their visit of a web-page. This is leading to serious limitations in terms of the allowed time for our system to produce its recommendations towards a specific visitor [24, 1]. Our analysis based on JavaScript functions deployed prior to the launch of the recommendation system has shown that the time window for the recommendations to be served is approximately 1 second, before the visitor reaches the part of the web-page at which the results are shown. This is making the time allowed for computation very strict in order for the recommendations to be seen and possibly clicked by the user.

The above limitation, lead to the system developed, performing demanding computational work offline, using cron scripts. The computations considered heavy to be performed real-time are:

- Producing cosine similarities between new and existent articles.
- Backing up and deleting data stored on the main database of the system, that are no longer useful for the recommendation engine.
- Clearing data from tables of the database that are considered noise.

Except for the data that are considered noise, all of the data deleted are first inserted into the cumulative tables used, and later backed up for possible future use. Much of the related work is focused or evaluated on offline systems [25, 22, 24, 4, 5, 18], which differs from the current work, since it combines real-time and offline characteristics.

Our proposed and developed system, combines characteristics from both collaborative and content-based filtering (both item and user-based), this shows similarities with parts of other work done [10, 19, 9, 27], in terms of combining techniques for the final result, however differs in the criteria used for the production of both content-based as well as collaborative filtering.

Some of the related work [10, 12] is focused on recommendations for search engines using appropriate techniques. However due to the difference in nature of athensvoice.gr and search engine websites, similarity between queries would be very inefficient to use in our case.

In order to assess the results produced to visitors by the recommendation system, apart from the data provided by the Google analytics service, a custom method was developed. Many assessment metrics have been proposed by the related work [25, 6, 12]. All these propositions, use clicks in order to evaluate the results of the recommendations. Our proposal differs than others suggested, since apart we also keep track of the criteria by which the click was produced. The above leads to a much more comprehensive analysis over the recommendations served to users of athensvoice.gr and also opens up a possible future integration of machine learning approaches within the recommendation system.

Similarly to most of the work done in this field, our system does not use user-ratings to produce the results, but visitor clicks instead. It is also used on top of a Drupal system, as the one already in use by athensvoice.gr.

It should be mentioned at this point, that our system's recommendations are solely based on the data it has collected, over the time period it has been used, at which it is storing data and cumulating it into the appropriate tables on the database. Since its launch, our system has been making recommendations towards the visitors of the website, while training at the same time. The above is making it different than other work done [27, 21, 18], since it did not use existent datasets, or data accumulated prior to its launch for training.

Another very important factor to take into account when designing such a system, is the cold start problem. The cold start problem, is the one at which a new user has visited the website, making a recommendation request to the system which has to serve its results back. Such a scenario is a problem for personalized content-based systems, since there is no data at that point to produce results. There has been much work on this field addressing this problem [18, 26], however it differs from our approach because user-based filtering is only one of the 5 criteria used in order to produce recommendations. So in case of a cold start situation, the rest 4 criteria kick-in to provide the results based on them.

There has also been work based on the factors to take into account, and the data to store in order to produce the best fitted recommendations for users. There has been work about cumulative data [21], location based techniques [17, 15], personalized [18, 4, 1, 16, 27] use of social services [13, 16]. However these works differs than this one, since the data stored are both per user (non-cumulative), cumulative and context and time aware.

Concluding this section, the main differences of the current work with other done on the same field, can be summarized to the following:

1. Handling of different genres of articles.
2. Training, running and assessment done on an a live website, with real-time visitors.
3. Combining online, real-time and off-line features (cron scripts).
4. Serving on the fly recommendations that lead to strict time allowed for computation.
5. Combining techniques and criteria for the recommendation algorithm.

6. Developing and using in production metrics and corresponding tools for the continuous assessment of the deployed system.

4. HIGH LEVEL DESCRIPTION OF RECOMMENDATION SYSTEM

In this section we describe the recommendation system developed in an abstract way, in order to show the flow of data, during its use. As presented in Figure 1, the complete system is separated into 4 sub-systems/servers:

1. The athensvoice.gr server for Desktop/Tablet users (S-DT)
2. The athensvoice.gr server for Mobile users (S-M)
3. The athensvoice.gr database server (S-DB)
4. The recommendation system server (S-RS)

Each of these servers has a specific role in the system, as we will see next.

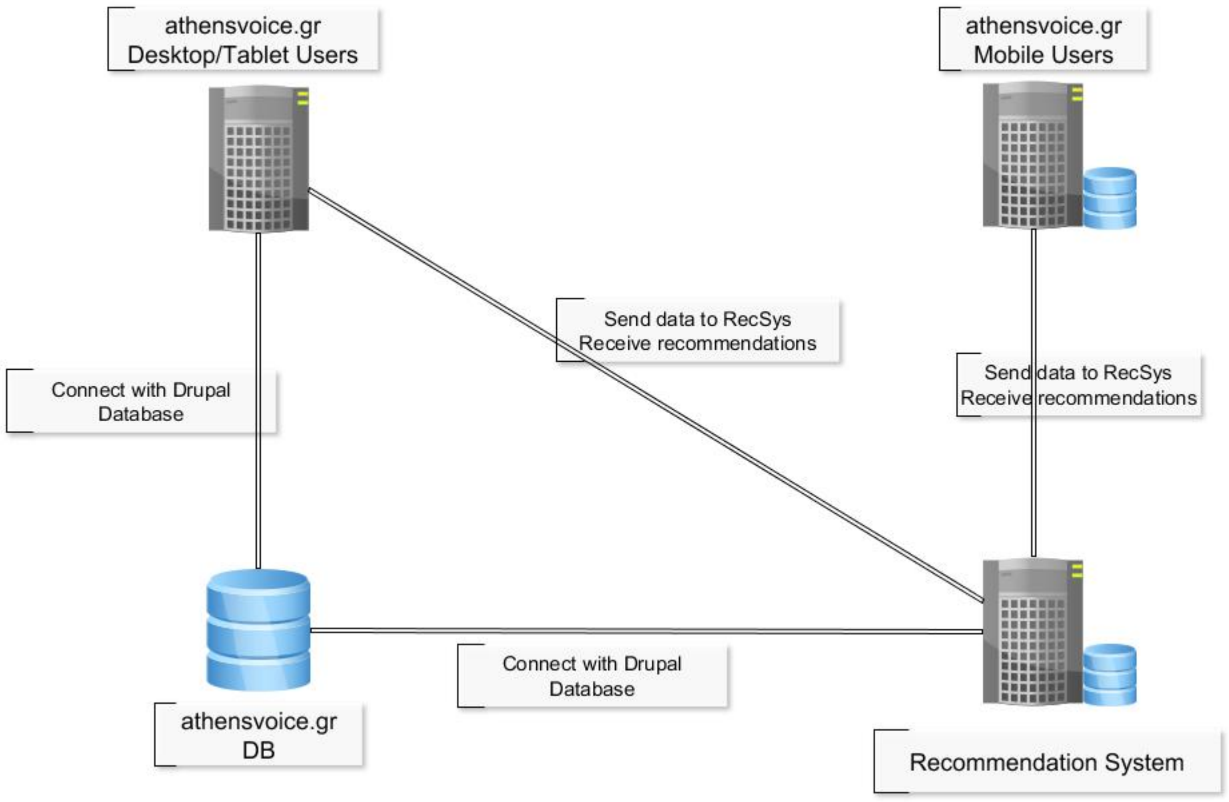


Figure 1: High Level System Diagram

First we have the athensvoice.gr server responsible for desktop and tablet visitors of the website. This is the server on which part of the Drupal system is deployed (apache and php scripts), that in turn communicates with the database server, in order to fetch the results of each query.

Next we have the athensvoice.gr server which handles requests made by mobile devices. This server has its own mirror of the main database which is synchronized with the main one, in a master-slave architecture.

Finally there is the web server, which hosts the hybrid recommendation system that was developed. At this server there are the php scripts necessary for the system to work, a mysql database in which data is stored and the cron scripts doing maintenance work.

It should be mentioned at this point that all of the servers are using CentOS Linux 6.6, communicate with each other through an internal LAN, and connect with the web using 1000Mbps network lines.

Algorithm 1 Algorithmic Overview of RecSys

```
 1: procedure User Visit
 2:     if visitor uses (desktop OR tablet) then
 3:         server_used ← S-DT
 4:     else
 5:         server_used ← S-M
 6:     currentURL ← current visited page
 7:     if first time visitor then
 8:         recSysID ← generate new user_id
 9:     else
10:         recSysID ← read recSysID value
11:     server_used makes request to S-RS with recSysID and currentURL
12:     S-RS stores data_for_visit and returns recommendations
13:     server_used shows recommendations to visitor
```

Figure 2: Algorithmic Overview of RecSys

The algorithm below, shows the steps taken during a user's visit on a web-page of athensvoice.gr. Initially, he/she is redirected to the appropriate server the handle his/hers request based on the device he is using. Then the id for the recommendation system is retrieved. His/hers id is passed to the recommendation system along with the current url of the visit. The recommendation system, stores the data derived by this information, and returns the recommendations based on it, and on the data already stored. These recommendations are presented to the visitor, on the predefined space, by the web-server handling the initial request.

5. TECHNICAL OVERVIEW OF RECOMMENDATION SYSTEM

In this section we describe the system developed in a more detailed way. In order for it to be easier understood, we have divided the description of the system in the following:

- Drupal system.
- Interface sub-system.
- Recommendation system database overview.
- Recommendation engine.
- Back-end sub-system.

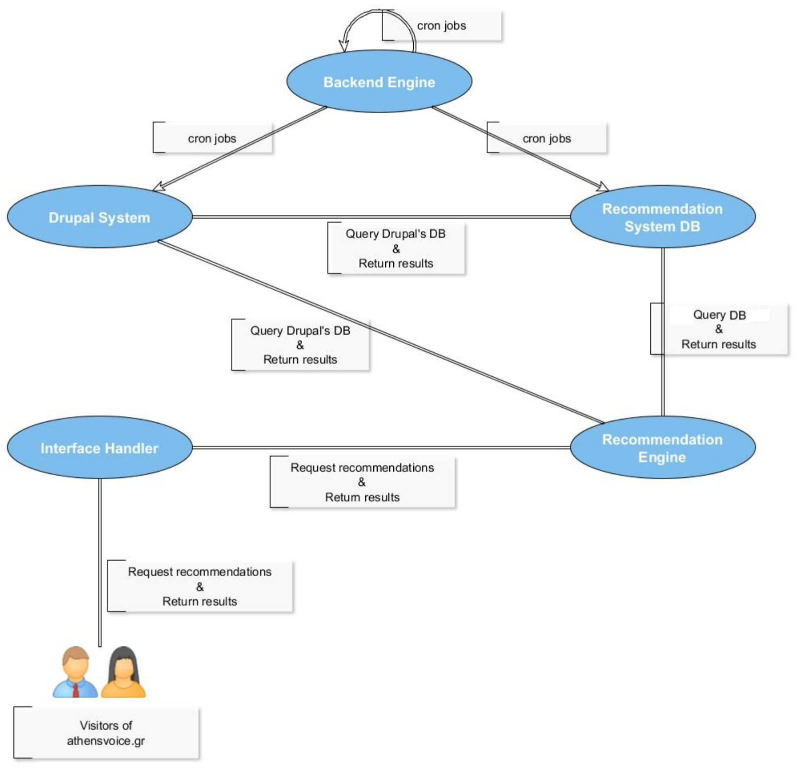

cron jobs
Backend Engine
cron jobs
cron jobs
Drupal System
Recommendation System DB
Query Drupal's DB & Return results
Query DB & Return results
Query Drupal's DB & Return results
Interface Handler
Recommendation Engine
Request recommendations & Return results
Request recommendations & Return results
Visitors of athensvoice.gr

Figure 3: Recommendation System Overview.

5.1 Drupal System

As mentioned previously, the developed recommendation system, was build on top of the drupal content management system (CMS), on which the website of athensvoice.gr is implemented. To be more specific, the version used is 6, and a number of modifications have been done, concerning mostly the storage engine of tables of the database it maintains.

Drupal is an open-source CMS, equivalent to Wordpress and Joomla, which allows a higher amount of freedom for developers to interfere with its core, compared with the other two.

This system depends greatly on 2 terms:

- Node
- Taxonomy

Node has to do with anything uploaded to the website (article, page, etc.). All of this data is uniquely identified by an integer, nid.

Taxonomy, is used for the categorization of content. A unique integer is given by the system for each taxonomy, tid.

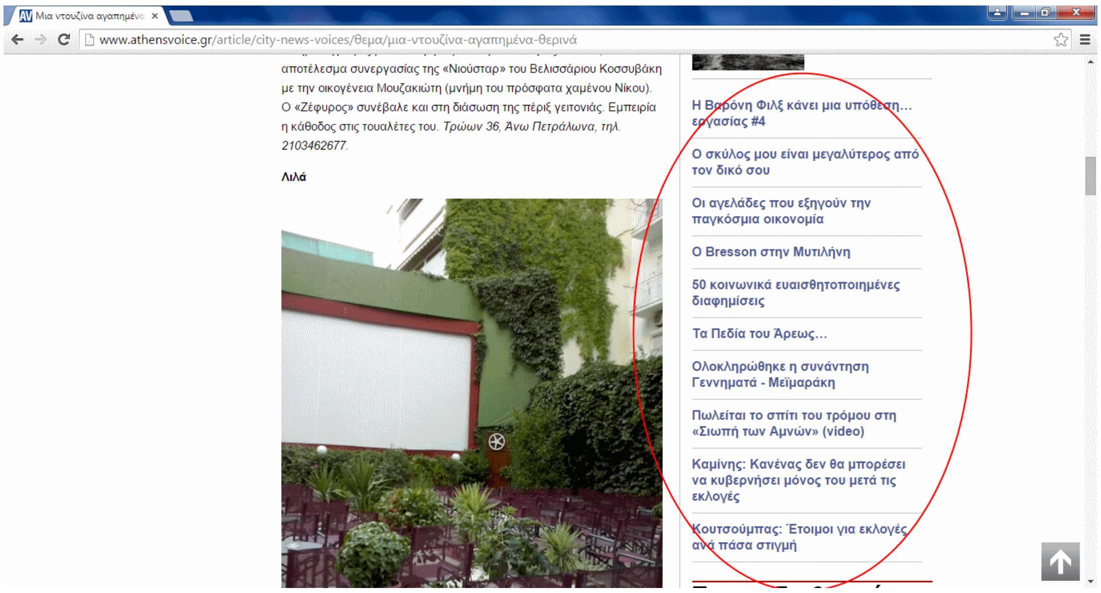
Both of the above integers, are used by the recommendation system, for its connection with the Drupal one, and are inserted into many of the tables of the database, as we describe later on.

5.2 Interface Subsystem

Interface sub-system is responsible for the presentation of the recommendations produced by the system, to the visitors of the website. To that end, the maximum number of recommendations towards the users has been set to 15. These recommendations are divided as follows:

- 4 recommendations come from the specific user's interests.
- 3 recommendations come from the relation between categories (taxonomies).
- 3 recommendations are based on the time of day.
- 3 recommendations are calculated using the cosine similarity between the current article read, and the rest available.
- 2 recommendations come from the most read articles for the current day.

All of the above recommendations produced, are first checked for possible duplicates before they are presented to the visitor.


Μια ντουζίνα αγαπημένα

www.athensvoice.gr/article/city-news-voices/θεμα/μια-ντουζίνα-αγαπημένα-θερινά

αποτέλεσμα συνεργασίας της «Νιούσταρ» του Βελισσάριου Κοσσυβάκη με την οικογένεια Μουζακιώτη (μνήμη του πρόσφατα χαμένου Νίκου). Ο «Ζέφυρος» συνέβαλε και στη διάσωση της πέριξ γειτονιάς. Εμπειρία η κάθοδος στις τουαλέτες του. *Τρώων 36, Άνω Πετράλωνα, τηλ. 2103462677.*

Λιλά

Η Βαρόνη Φιλξ κάνει μια υπόθεση… εργασίας #4

Ο σκύλος μου είναι μεγαλύτερος από τον δικό σου

Οι αγελάδες που εξηγούν την παγκόσμια οικονομία

Ο Bresson στην Μυτιλήνη

50 κοινωνικά ευαισθητοποιημένες διαφημίσεις

Τα Πεδία του Άρεως…

Ολοκληρώθηκε η συνάντηση Γεννηματά - Μεϊμαράκη

Πωλείται το σπίτι του τρόμου στη «Σιωπή των Αμνών» (video)

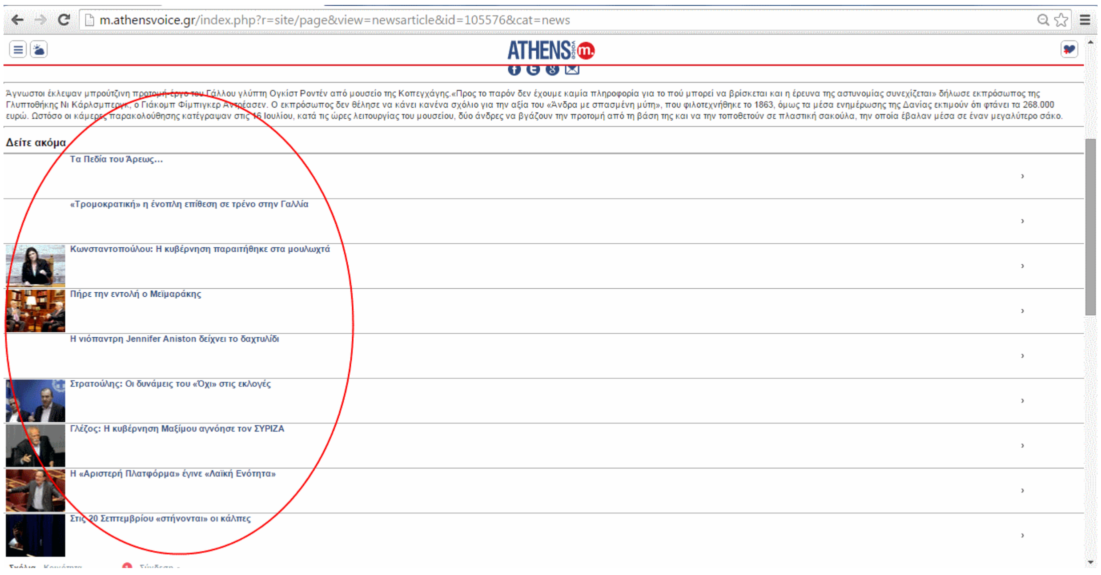
Καμίνης: Κανένας δεν θα μπορέσει να κυβερνήσει μόνος του μετά τις εκλογές

Κουτσούμπας: Έτοιμοι για εκλογές ανά πάσα στιγμή

Figure 4: Recommendation's area for desktop/tablet visitors.

Furthermore, we chose to present the recommendations differently, based on the device a visitor of the website uses. Figure 4, shows the area and format of the recommendations to dektop and tablet users. The area and format of the recommendations made towards mobile device users is shown in figure 5. The above distinction was made due to the following reasons:

- Based on the device a visitor uses, the web page changes.
- Requirements (in terms of: connection speed, screen size, etc) change based on the device used.


m.athensvoice.gr/index.php?r=site/page&view=newsarticle&id=105576&cat=news

Άγνωστοι έκλεψαν μπρούτζινη προτομή-έργο του Γάλλου γλύπτη Ογκίστ Ροντέν από μουσείο της Κοπεγχάγης.«Προς το παρόν δεν έχουμε καμία πληροφορία για το πού μπορεί να βρίσκεται και η έρευνα της αστυνομίας συνεχίζεται» δήλωσε εκπρόσωπος της Γλυπτοθήκης Νι Κάρλσμπεργκ, ο Γιάκομπ Φίμπιγκερ Ανδρέασεν. Ο εκπρόσωπος δεν θέλησε να κάνει κανένα σχόλιο για την αξία του «Άνδρα με σπασμένη μύτη», που φιλοτεχνήθηκε το 1863, όμως τα μέσα ενημέρωσης της Δανίας εκτιμούν ότι φτάνει τα 268.000 ευρώ. Ωστόσο οι κάμερες παρακολούθησης κατέγραψαν στις 16 Ιουλίου, κατά τις ώρες λειτουργίας του μουσείου, δύο άνδρες να βγάζουν την προτομή από τη βάση της και να την τοποθετούν σε πλαστική σακούλα, την οποία έβαλαν μέσα σε έναν μεγαλύτερο σάκο.

Δείτε ακόμα

Τα Πεδία του Άρεως…

«Τρομοκρατική» η ένοπλη επίθεση σε τρένο στην Γαλλία

Κωνσταντοπούλου: Η κυβέρνηση παραιτήθηκε στα μουλωχτά

Πήρε την εντολή ο Μεϊμαράκης

Η νιόπαντρη Jennifer Aniston δείχνει το δαχτυλίδι

Στρατούλης: Οι δυνάμεις του «Όχι» στις εκλογές

Γλέζος: Η κυβέρνηση Μαξίμου αγνόησε τον ΣΥΡΙΖΑ

Η «Αριστερή Πλατφόρμα» έγινε «Λαϊκή Ενότητα»

Στις 20 Σεπτεμβρίου «στήνονται» οι κάλπες

Figure 5: Recommendation's area for mobile device visitors.

5.3 Recommendation System Database Overview

Storing the data acquired by the recommendation system is handled by its database sub-system. To cover the database sub-system need, 2 schemas are used:

- tracker
- backup_initdata

The first one, is where the tables used for the recommendations as well as the data for each visit is stored, and consists of 13 tables. The second one is where unused data, due to the their old timestamp, is stored.

5.3.1 Tracker Schema

As mentioned above, this schema is where the data used for recommendations and tracking of each visit is stored. It is the schema mainly used by the system developed, and consists of the following tables:

- Table initdata, which is where all the primitive data concerning a specific visit are stored.
- Table user_from_to, which associates categories for each visitor by tracking every click made.
- Table user_has_read, which stores data concerning the id of the article and the id of its category for each visitor.
- Table user_from_nid_to_nid, which stores the data concerning the flow of each visitor from article to article within the website of Athens Voice.
- Table user_profile, which stores cumulative data for each visitor that concern his/hers reads of a specific category.
- Table time_tid, which associates for each visitor, the category he/she reads, and the timezone the read occurred.
- Table time_profile, which stores cumulative data associating category reads with a specific timezone.
- Table tid_profile, which stores cumulative data associating each category of the website with another.
- Table nid_profile, which stores the number of times two articles were clicked successively, and the cosine similarity between them.
- Table cosine_sim, which stores the cosine similarity value between 2 specific articles of the website.
- Table day_tid, which associates each visitor id, the category he/she reads and the day of the week he/she reads it.

- Table day_profile, which stores cumulative data associating each category, day of the week and the amount of times the specific category was read on that day.
- Table node_counter, which is a Drupal maintained table and includes the information of day reads of each article of the website.

The above described tables, are the ones actively used each day by the system developed, in order for the recommendations to be produced, based on the algorithm we describe in a later section. Figure 6, shows the complete architecture of the database used by the recommendation system.

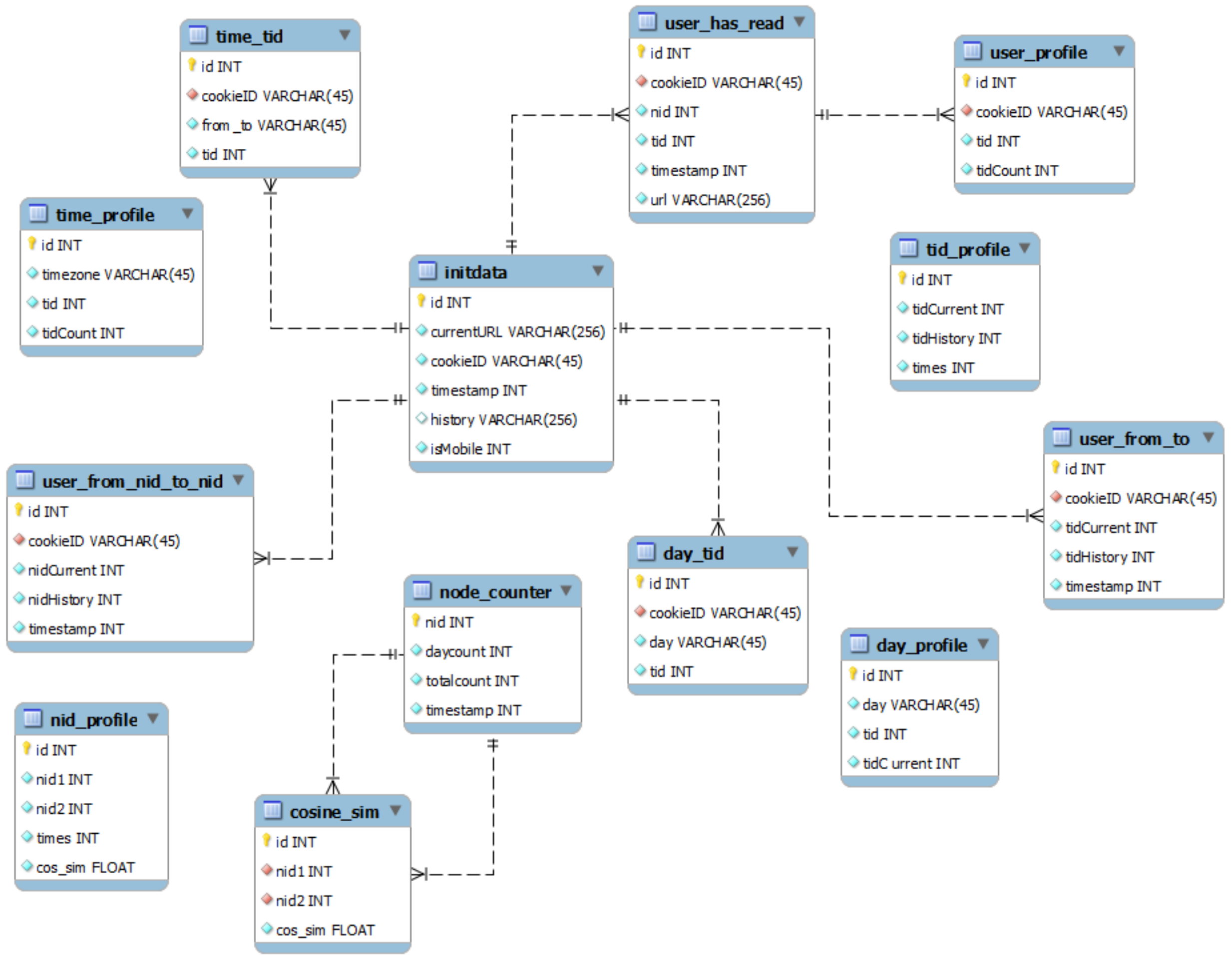


Figure 6: Complete database.

As can be seen by figure 6, the main table of the recommendation system's database is the initdata table. This table is where all the raw data collected by the system is inserted, and after proper recalculation this data is inserted into the other tables of the main schema.

The cookieID field is where the unique id of each visitor is stored. This field is the most important one in the database, since it is used in most of the tables as a means to identifying the visitor's preferences and behaviour throughout the website of athensvoice.gr. The cookieID field is used into the following tables:

- Table time_tid in order to show each visitor's preferences during a specific time of day.

- Table user_has_read in order to determine the articles each visitor has read and not show them again.
- Table user_profile in order to find a visitor's favourite categories.
- Table user_from_nid_to_nid in order to know the sequence of each visitor's clicks throughout the website, which if used in collaboration with the nid_profile table can tell us his/hers favourable cosine similarity between the articles read.
- Table user_from_to in order to find the categories a visitor clicks sequentially.

The currentURL field is used in collaboration with Drupal's database in order for the article's nid and respective category's tid to be calculated. This data is then inserted throughout the database's tables that that require it. Such tables are:

- Table time_tid, into the tid field.
- Table user_has_read, into fields nid and tid.
- Table user_profile, into the tid field.
- Table time_profile, into the tid field.
- Table tid_profile, into the tidCurrent field.
- Table user_from_nid_to_nid, into the nidCurrent field.
- Table user_from_to, into the tidCurrent field.
- Table day_tid, into the tid field.
- Table day_profile, into the tid field.

The history field is handled in a similar fashion to the currentURL one. In this field information about the previous article's nid and category tid is stored. After communicating with Drupal's database and fetching this data, we then insert it into the following tables:

- Table tid_profile, into the tidHistory field.
- Table user_from_nid_to_nid, into the nidHistory field.
- Table user_from_to, into the tidHistory field.

The recommendation algorithm we have developed is based on 5 criteria for the recommendations to be calculated. These 5 criteria depend on 6 tables of the database.

The user_profile table is where the first criteria is based on. This table stores the data concerning the favourable categories of each visitor of the website. It associates each cookieID with each category read and the number of times (tidCount field) these reads have occured. Using the data on this table we can present the visitor with articles from his/hers favourite categories.

The tid_profile table is where the association of categories of the website is stored. In this table we have the information about the frequency each category is accessed directly after another. Using the data from this table we are able to show the visitor articles that are in a category usually clicked after the category of the article he/she is currently reading.

The tables time_profile and day_profile is where we store information associating the time of day and day of the week with the categories read respectively. These 2 tables are used together by the recommendation system for it to make its recommendations. Using the data stored into these tables we present the visitor with articles that have a category usually read on a specific day of week and time of day.

The node_counter table, which is a table maintained by Drupal, is where statistics concerning each article of the website is stored. Using this table we show the visitor articles that are most frequently read by others on the current day.

The cosine_sim table is where each article is associated with another, along with their cosine similarity value. As we will further describe in the next section, this is a table populated using a cron script. Using the data stored into this table our system suggests content that have the maximum cosine similarity value with the one he/she currently reads.

In the recommendation system we have developed we chose to not present the visitor with content already read by him/her. This is done for the following reasons: content already read by a visitor would not interest him/her any more, such content can be searched using the tools available on the website and articles read are already presented to him/her in various spots on each web-page. In order for us to be able to exclude recommendations already read by the visitor, we introduced the user_has_read table. In this table we store for each visitor the articles he/she has read along with when this read has occurred. Using the data stored on this table our system recommends to visitors content that they have not already read form the website.

5.3.2 Backup Schema

The schema backup_initdata is where the system's data that are no longer useful real-time by the recommendation system is stored. In order to decrease the size of the tables and as a result increase the performance of the system, it was chosen that such data are first inserted into this backup schema, and then deleted from the one described in the previous subsection. The inserts and deletes are handled by cron scripts that run on the server. All of the tables of this schema are in the format below:

- initdata{from}_{to}
- user_has_read{from}_{to}
- time_tid{from}_{to}
- day_tid{from}_{to}

Where {from} and {to} depict the time range that the inserted data cover. As we will explain in more detail into a later section, the data considered obsolete and as a result inserted into the respective tables may be due to one of the following reasons:

1. Data from the initdata, time_tid and day_tid tables are considered useless at the end of each week. This data is stored for usage by possible expansions of our current work and statistical purposes.
2. Entries into the user_has_read table are not considered important if they are timestamped 10 days ago. This is done since our recommendation algorithm does not go before that period of time in search of suggestions towards visitors.

5.4 Recommendation Engine

The recommendation engine is the part of the system that given a specific input, returns the suggestions towards a specific visitor. It utilizes the system's database and is the one running between the database and the interface.

Given as input the unique identifier for the visitor (cookieID), and the current url he/she is viewing, the engine returns 15 suggestions. We came to this number of total recommendations served based on the following parameters:

- The slot on the web-page reserved for these suggestions could not exceed this limit, since it would make other slots of the page appear at a lower position.
- The total number of recommendations per criteria was chosen to be on average 3 per criteria with the exceptions of user profile and most read criteria which serve 4 and 2 recommendations respectively.

This number of 15 recommendations towards the visitor are generated based on the criteria described below:

- From the table user_profile, a specific visitor's favourite categories are drawn (4/15).
- From the table tid_profile, articles are selected using the correlation between their categories (3/15).
- From the tables day_profile and time_profile, are used to fetch articles based on the day and current timezone (3/15).
- From the table node_counter, articles are chosen that have been read the most on a specific day (2/15).
- From the cosine_sim table, articles are selected that are closer to the current one being read by a visitor (3/15).

The above leads to 5 criteria for the recommendations made by the system. More specifically, the first criteria, which is based on the user profile built by the system, makes it a user-based one. This criteria, generates suggestions based on each visitor's specific interests and behaviour during his/hers visit on the website.

The second one (from the correlation of the categories), makes it a collaborative system, since the information used is based on the visits of every user of the website. The recommendations served based on this criteria, rely on the most probable category read immediately after the current one each visitor is reading.

The third one, that takes into account the day and time for the results chosen, combines characteristics of a collaborative system and a context aware one. This criteria makes its recommendations using the frequent article categories read during each day of week and time of day.

The fourth criteria, that uses the number of today's reads of an article, is giving the system the notion of trend to take into account. It serves its recommendations based on the most read articles of the current day, filtering out those the visitor has already read.

Finally the fifth criteria, that uses the cosine similarity of the current article being read and the next available, is making the system developed a content based (using items) one. The suggestions produced based on this criteria are the articles that have a higher cosine similarity value with the current one being read by the visitor.

It should be mentioned at this point, that from all of the recommendations produced by the system, the ones a visitor has already read are excluded, and prior to being served back at the user are checked for possible duplicates. As mentioned before, the total number of recommendations served by each criteria was initially chosen to be 3 recommendations per criteria. This was made in order to be as objective as possible in trying to evaluate each criteria used by the recommendation system developed. However due to the fact that on various slots of each web-page of athensvoice.gr the number of an article's reads is used, we chose to show 1 less recommendation from the most read criteria and show 1 more from the user profile one.

5.5 Backend Engine

In this section we describe all of the features of the backend engine, which is part of the recommendation system built. This subsystem is responsible for the offline, maintenance work performed on the recommendation system. It does most of the heavy, in terms of calculation, work as well as cleaning of the main database from data that are no longer useful.

The work performed by the backend engine, is in the form of php scripts, that run on the server hosting the recommendation system as cron jobs. These processes are:

- Process reset_user_profile, is the one cleaning the table user_profile, from data that are considered noise. These data cover a specific visitor's reads of a category that is below 3, thus making it impossible for the system to produce accurate recommendations. The data are inserted into the backup schema, before their deletion and the process is run every 15 days.
- Process reset_initdata, is performing cleaning tasks on the following tables:
  - initdata, empty the table
  - time_tid, empty the table
  - day_tid, empty the table
  - user_has_read, removing data stored 10 days ago

  This process is run once every week, and before deleting data from the main database, it inserts them into the backup schema described on a previous section.

- Process reset_uhr_noise, resets the user_has_read table from data considered noise. Since the recommendation system depends on JavaScript for the initialization of a specific visitor, it is possible (although rare) that he/she has disabled such a feature from the browser's settings. This inaccurate data being stored into the user_has_read table is deleted by the reset_uhr_noise process, which runs every day.
- Process produce_cossim, updates the nid_profile table, with the cosine similarities of articles that has not yet been calculated. Because such a task is computationally heavy, it is performed as a cron job that runs every 20 minutes.
- Process insert_cos_sims, is the one inserting data into the cosine_sim table of the main database. This process is also run every 20 minutes.
- Process pl_problem_solver, aims in solving a common problem in recommendation systems that are either content based or collaborative, the stability vs plasticity one [14]. This problem occurs when after some time, the user's preferences are difficult to change, given the fact that a number of visits of the specific user have been documented. In order to prevent that behaviour in our system, the pl_problem_solver process is run every 15 days. This process reduces the counters stored in cumulative tables by 25% for those values that they have a big difference (>80%) with other rows of the equivalent tables.
- Process run_queries, executes the update queries stored into a file, at the end of every day. It was observed that during hours of heavy traffic for athensvoice.gr, the recommendation system was returning results to the visitors slower than usual. To remedy that we chose to separate insert queries (which are an important real-time feature of the system), with the less critical ones, which are update queries. These update queries are stored into a file, accessed by this process.

5.6 Custom Metric Description

As part of this work, a novel approach for evaluating of the effects of the recommendation system was developed. Due to the fact that this system is a hybrid one, offering its recommendations using a number of criteria, the necessity to track the criteria by which a visitor's click on a recommendation occurred, emerged. To cover this need, we developed a custom way of tracking this click, which we describe in this section.

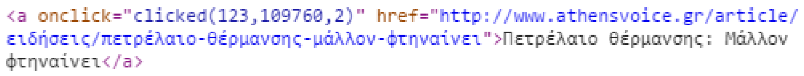
This custom metric uses the visitor's click as the main factor for the evaluation of the recommendation algorithm. Along with the user's click we also keep track of the criteria by which the click was produced. This is used in order to assess the value of each of the criteria deployed in the recommendation algorithm.

For us to be able to track the user that clicked a recommendation as well as the criteria by which the recommendation was produced and the timestamp of the click, 2 new tables were inserted into the database of the system developed. These 2 tables are identical in terms of fields, and only differ as to where the records of each one came from. We distinguish these 2 tables by the device a visitor uses for his/hers browsing of Athens Voice.

As we will describe in a later section in more detail, the recommendation algorithm deployed in the system utilizes 5 criteria to make recommendations towards visitors. These 5 criteria are the following:

- Criteria 1 is the user's profile based on the categories he/she reads when visiting the website.
- Criteria 2 is the category's profile based on the association between each one in the website.
- Criteria 3 is the day and time profile based on the categories most frequently read on a specific day of the week and time of day.
- Criteria 4 is based on the articles most read on a specific day.
- Criteria 5 utilizes the cosine similarity between 2 articles.

In order to track the clicks of a user a JavaScript function is attached to the onclick event of every recommendation served by the system. This function takes as input the id of the user, the id of the article of the specific recommendation and the criteria by which it was produced. It sends this data to the server that hosts the recommendation system which in turn inserts these values along with the current timestamp into the respective table of the database. An example of this addition to the recommendations can be seen in figure 7.



```
<a onclick="clicked(123,109760,2)" href="http://www.athensvoice.gr/article/
ειδήσεις/πετρέλαιο-θέρμανσης-μάλλον-φτηναίνει">Πετρέλαιο θέρμανσης: Μάλλον
φτηναίνει</a>
```

Figure 7: Example of onclick JavaScript function

This method leads to 2 very useful insights for the recommendation system. By tracking the clicks of each user, we can evaluate the use of our system in the website and be able to see the amount of its positive effect in terms of CTR. Furthermore, by keeping track of the criteria by which each click was produced, we are able to evaluate the effect each criteria has on the system. As mentioned in a previous section, this leads to possible integration of machine learning techniques within the recommendation system developed.

6. RECOMMENDATION ALGORITHM

In this section, we describe the recommendation algorithm developed for the system. We start with an overview of the data it collects and the way it is stored in both the system's database, as well as the visitor of the website. We go on to analyse their usage for the system internally, and finally we describe how this data is evaluated, in order for the recommendations to be produced and returned to the visitor of athensvoice.gr.

6.1 System Data Overview

In this subsection, we will make a more analytical description of the data of the system as to how this data is produced and how it is stored internally into the database of the recommendation system.

In order for the visitor of the website to be uniquely identified by our system, a unique id needs to be produced (cookieID). This is accomplished through a JavaScript function running on the user's end, every time he/she is viewing a web-page. This function initially checks the user's browser for the existence of a specific cookie. If it already exists, the function renews its expiration date to one year in the future and returns the value. If it does not exist, the function creates the id for the user and sets it in a cookie stored at the user's browser for future use.

The recommendation system also uses 2 url associated data:

- current url
- previous url

The above 2 urls are initially used by the system, for their insertion at the respective table and after the communication with the Drupal system, they are translated into usable ids for both the article and its category they designate. The returned data by the Drupal systems along with the id of the specific user, are then used for the insertion and update of the cumulative tables they affect.

Other data also used by the developed recommendation system are:

- day of the week
- time zone of the day

This information is used by the system, in order for the association of categories and day and time to be produced. This feature is giving our system context awareness. As mentioned in a previous section, each day is divided into 5 time zones:

- 02:00-07:59
- 08:00-12:59
- 13:00-16:59

- 17:00-20:59
- 21:00-01:59

Another set of data that is being used, is the cosine similarities between articles posted on athensvoice.gr. This is done by cron jobs running on the server that hosts the system, and is exploited for the association of articles.

Algorithm 2 Algorithmic Overview of Data Handling

```
1:  procedure User Visit
2:      currentURL ← current visited page
3:      historyURL ← previously visited page
4:      if first time visitor then
5:          recSysID ← generate new user_id
6:      else
7:          recSysID ← read recSysID value
8:      RecSys renews cookie expiration date +1year
9:      currentURL, historyURL, recSysID are sent to RecSys
10:     RecSys communicates with Drupal to receive drupal_data
11:     RecSys updates tables using currentURL, historyURL, recSysID,
        current_timestamp, drupal data
12:     RecSys produces recommendations for the user
13:     RecSys returns recommendations
```

Figure 8: Algorithmic Overview of Data Handling

6.2 Data Usage

In the previous section, we analysed the data that the system collects. In the current one, we focus on describing the way these data are used, in order for the final recommendations to be produced.

The recommendations served by our system are based on the following criteria:

1. The user's profile criteria, using information stored for each visitor's categories and articles read, presents the visitor with articles from categories he/she prefers. Out of a total of 4 recommendations:

    - 2, come from the user's most read categories
    - 2, from the rest read by the user

   From the most read categories of the visitor, 2 recommendations are fetched that the visitor has not seen during the past 3 days. The rest 2 recommendations from this criteria, are chosen from categories the visitor reads, but are not below 20% of the most read. The articles chosen in this way may go back up to 6 days.

2. The current category read criteria, using information stored about the correlation

between categories of the website, retrieves articles from categories that are usually chosen by visitors and have a strong relation with the current one read. Using this criteria 3 recommendations are produced, excluding though the current one read by the visitor, since due to the organization of each web page of athensvoice.gr he/she is already presented with such choices.

3. The day and time of visit criteria, using the data associating the day of week and time of day with the frequent categories read, recommends content from categories usually accessed on the specific time and day. From this criteria 3 recommendations are produced and presented to the visitor, giving the developed system context and time awareness.
4. The most read criteria, using the current day's counters for reads, presents the visitor with articles that are frequently being read on that specific day. This empowers the system with the sense of trend for each day.
5. The cosine similarity criteria, using the cosine similarity value between the current article being read and other available, suggests each visitor with articles that show a high similarity with the current one read. This criteria produces 3 more recommendations, which similarly to the rest criteria, excluding the ones already read by the visitor.

6.3 Data Evaluation, Final Recommendations

During the past subsections we gave an overview of the data the system stores and uses, and then we described the way this data is being used, in order for the 5 criteria on which to base recommendations to be utilized. In this subsection we describe the way this data is being evaluated and possibly further filtered so that the final recommendations to be produced.

As mentioned on a previous section, although small, there are users of the website that may have disabled cookies and/or JavaScript on their browsers. This leads to either null entries as the id of each user or wrong data being inserted into the initdata table of the system. Since these data may lead to inaccurate recommendations, as far as the user profile goes, it was chosen to be deleted from the main system tables.

Among the information being stored by the recommendation system, is that of the previous web page a visitor was viewing. As can be easily understood, this may be within the athensvoice.gr bounds, or from external references, such as:

- google searches
- social media networks (facebook, twitter, etc)
- other sources

As described in another section, the system stores the id of the previous page the visitor was on. In order for that to be made possible with external sources, the following matching was performed:

- facebook:-1
- twitter:-2
- google:-3
- other sources:-4

This makes it possible for the association between current and previous categories to remain unchanged as a process, without having to make a special case if the visitor was previously in athensvoice.gr or another website.

As mentioned in the section concerning the backend engine of the system, measures are taken against the plasticity problem. This is done through cron jobs, and makes it possible for association between categories as well as categories that a specific visitor favours, remain unchanged by categories that are read only because they are popular for a particular time period.

For the final recommendations to be produced, 5 criteria are being used:

- the user's profile (4/15)
- the association of different categories (3/15)
- current day and time (3/15)
- cosine similarity between the current and other articles (3/15)
- most read articles of the day (2/15)

From the above criteria a maximum of 15 recommendations are returned, that do not include articles already read by the specific visitor. Due to the fact that it is possible that a criteria does not return its maximum amount of articles, the difference is passed down to the next criteria. For example, if from a specific user's profile, instead of 4, 2 recommendations are returned, the remaining 2 are passed to the category profile criteria, from which 5 recommendations are expected and so on.

Algorithm in figure 9, shows the recommendation algorithm in an abstract algorithmic way.

6.4 System Complexity Overview

In the previous sections we described the recommendation algorithm in terms of the data it stores, the way this data is being used and how they are evaluated in order for the final recommendations to be produced. In this section we perform a complexity analysis of our recommendation system. We divide this analysis in 2 axis: one based on the script running, and another based on the time elapsed for the execution of each part of the script. We chose to do this analysis on the following 3 scripts, which are the ones performing the

Algorithm 3 Algorithmic Overview for Recommendations Produced

```
1: procedure Produce Recommendations
2:     $maxUP \leftarrow 4$
3:     $final\_recommendations \leftarrow \{empty\}$
4:     $recommendationsUP \leftarrow$ maxUP recommendations from user profile
5:     $maxCP \leftarrow 3$
6:     if count($recommendationsUP$)<$maxUP$ then
7:         $maxCP+ \leftarrow (4 - maxUP)$
8:     $recommendationsCP \leftarrow$ maxCP recommendations from category profile
9:     $maxTDP \leftarrow 3$
10:    if count($recommendationsCP$)<$maxCP$ then
11:        $maxTDP+ \leftarrow (3 - maxCP)$
12:    $recommendationsTDP \leftarrow$ maxTDP recommendations from time/day profile
13:    $maxCSP \leftarrow 3$
14:    if count($recommendationsTDP$)<$maxTDP$ then
15:        $maxCSP+ \leftarrow (3 - maxTDP)$
16:    $recommendationsCSP \leftarrow$ maxCSP recommendations from cosine similarity table
17:    $maxMR \leftarrow 2$
18:    if count($recommendationsCSP$)<$maxCSP$ then
19:        $maxMR+ \leftarrow (3 - maxCSP)$
20:    $recommendationsMR \leftarrow$ maxMR recommendations from most read of day
21:    $final\_recommendations \leftarrow$ distinct(recommendations of all criteria combined)
22:    return $final\_recommendations$
```

Figure 9: Algorithmic Overview for Recommendations Produced

heavier work:

- initdata.php, the script that stores the data for each visitor of the website, and his/hers preferences
- render.php, the script responsible with calculating and serving back the recommendations for each user
- run_queries.php, the script running as a cron job, that handles inserts and updates to the database that are not important to be done in real-time

6.4.1 initdata.php

In this section, we show the complexity of the initdata.php script, which handles the storage of data for each visit of each user of the website. We analyse the complexity of this script in terms of the time elapsed for its execution in 2 scenarios, the average one, in which the user has visited the website before and the association between the categories he reads has already been calculated, thus having fewer inserts and updates to be done towards the database in real-time, and the worst case one, in which the user has never visited the website before, and the data stored in the cumulative tables has never been calculated before so more queries than the average scenario have to be performed.

Using the big O notation, the complexity of this script is:

$$O(n)$$

where:

- n, is the number of commands performed

In order to track the time of the execution, we define the following:

$$TotalTime = t(includes)+t(query\_generation+file\_insertion)+t(query\_execution)+t(func\_calls)$$

where:

- $t(includes)$, is time needed for the necessary includes
- $t(query\_generation + file\_insertion)$, is the time needed for the script to generate the queries that are then inserted into the respective file to be performed through a cron job
- $t(query\_execution)$, which depicts the time needed for the execution of queries which are run real-time
- $t(func\_calls)$, is the remainder time elapsed for the execution of the script, and in which function calls are made

Figure 10, shows the time of execution (in milliseconds) for each part of the script in both average and worst case scenarios.

t()	**Average Case**	**Worst Case**
$t(includes)$	0-1ms	
$t(query\_generation + file\_insertion)$	0-15ms	0-10ms
$t(query\_execution)$	450-750ms	900-2700ms
$t(func\_calls)$	∼300ms	
TotalTime	750-1066ms	1200-3011ms

Figure 10: Execution time in milliseconds for initdata.php

As can be seen from the above table, in the average case scenario, the execution time of the script is about 1 second (varying from 0.7-1) whereas in the worst case one, it varies from 1.2-3 seconds if none of the queries can be cached to performed later.

6.4.2 render.php

Similarly to our work in the previous section, we analyse in this the complexity for the render.php script. This script is responsible for calculating and serving the recommendations to each visitor of the website. Again we use 2 scenarios, the best case one, where the visitor has never visited the website before and as a result, there are 2 operations less to calculate, and the average case one where he/she has already visited the website before and all of the criteria and calculations have to be done. The worst case scenario in this script is the average case one where all the criteria are in use.

Using the big O notation, the complexity of this script is:

$$\leq O(n^2)$$

Since there are nested iterations which at most will be executed without reaching the break statement.

For the tracking of the execution time of the script, we define the following:

$$TotalTime = t(fetch\_uhr\_articles) + t(recs\_generation) + t(remove\_duplicates) + t(render)$$

where:

- $t(fetch\_uhr\_articles)$, is time needed to fetch the article the current user has already read
- $t(recs\_generation)$, shows the amount of time needed to generate recommendations for the visitor using the 5 criteria of the system
- $t(remove\_duplicates)$, is the time for the removal of possible duplicates in the recommendation array
- $t(render)$, is the time needed for the rendering of data

Figure 11, shows the time of execution (in milliseconds) for each part of the script in both best and average case scenarios. As can be seen, the execution time of the script varies from 0.3-0.8 seconds. As mentioned in a previous section effort has been placed, in order to keep this time as minimal as possible, since the time window allowed for the recommendations to be seen by the visitor of the website is very strict.

t()	**Best Case**	**Average Case**
$t(fetch\_uhr\_articles)$	0ms	80-165ms
$t(recs\_generation)$	365-585ms	435-660ms
$t(remove\_duplicates)$	0-1ms	
$t(render)$	0-1ms	
TotalTime	365-587ms	515-827ms

Figure 11: Execution time in milliseconds for render.php

6.4.3 run_queries.php

The run_queries.php script is a script running as a cron job on the server hosting the recommendation system at the end of each day and executes the queries stored on a specific file, which even though useful for the system, do not need to be executed real-time.

Since in the file accessed by the script, INSERT and UPDATE commands are stored and executed sequentially, the complexity is:

$$O(n)$$

where:

- n, is the number of commands to be executed

In order to calculate the execution time of this script, we should note the number of commands stored in the file, which may vary from 1,041,530-1,060,924 which are the minimum and maximum values for the commands stored in the respective files during the week: 12/10/2015 - 18/10-2015. Having an average of 250ms for each query, the total execution time of this script is: 72.3-73.5 hours!

This result signifies accurately the importance of performing the non-crucial SQL commands offline, using 1 connection to the SQL server, which if performed real-time would severely increase the execution time of the initdata script, and as a result have a number of hanging connections to the server, which are not necessary. As mentioned on a previous section, we remedy this by storing update queries (and inserts into the initdata table) in a file, which is accessed by this script at the end of each day.

7. RESULT ANALYSIS

In this section we present the results of the use of the system at the website of athensvoice.gr. We start by a general overview of the statistics, and then we go on to analyse the results using the Google Analytics service. At that point we present the results when the system is in use as well as when the system is deactivated. At the next subsection, we describe the custom metric that was developed as part of this thesis, and we conclude the section by presenting the results as they were documented using this tool.

It should be mentioned at this point that the results presented in this section are concerning the following time periods:

- General statistics: 22/04/2015-22/08/2015
- Google Analytics data, with the system in use: 24/07/2015-24/08/2015
- Google Analytics data, with the system deactivated: 29/09/2015-30/09/2015, compared with the data of the previous week
- Custom metric developed: 10/09/2015-22/09/2015

The rest of this section is organized as follows:

1. We present the general statistics of the data stored into our recommendation system. This is done to show a general overview of the traffic of the website the system is used into, as well as to show the amount of data used by the system in order for it to have the results described.
2. We move on to analyse the results of the system's integration into the website of Athens Voice, using Google's Analytics service. This is done in order to both present the results using a third-party tool, as well as show the recommendation system's importance by deactivating it for a limited period of time and studying the results.
3. We describe the results of the system's integration using the custom metric we developed. This is done so that we show the total number of clicks made to recommendations served by our system. Using this metric we move on to analyse the performance of each criteria deployed within the recommendation system by taking into account the number of clicks each has offered.
4. We conclude this section by studying the benefits this system has offered by being integrated within the website of Athens Voice. This is performed by analysing 2 scenarios, best and worst case ones, for the clicks made to recommendations served by our system. These 2 scenarios are then analysed economically, in terms of average advertisement revenue, in order to present the reader with the economic benefits of this integration into a live website.

7.1 General Statistics Overview

In this subsection we present general statistics of the system and the data it has stored, that cover a 4 month period (22/04/2015-22/08/2015). The total data collected concern 27.718.535 user's visits, without counting the ones considered noise. From these data, stored in the initdata table, is where the rest of the system's tables were created.

The data from which the users profiles were created, came from these of the navigations of visitors between articles, that are: 10.393.173.

The category profiles that are stored by the recommendation system, are 13.693. These data have to do with the association of categories between them.

The data that are being used in order to exclude already read articles by the visitors of athensvoice.gr are: 19.188.534. These data are also used in order to populate other tables of the database of the system.

The unique users for whom data has been collected until 22/08/2015 are: 1.562.134.

The data that show the navigations of users from article to article, are: 401.885. Whereas the cosine similarities between articles that have been calculated are: 53.406.449. These 2 sets of data combined with these of the profile of articles (nid_profile, 126.740 records) are the ones with which the respective criteria makes its recommendations.

For the use of the day and time criteria, data from 4 tables are being accessed. Two of them concern the association between day, time and the categories being read. The amount of data stored in these tables is: 5.909.469 for the day and 6.494.281 for the time. From these tables the respective cumulative ones are produced which until 22/08/2015 have a total of 3.373 records inserted in them.

We should finally mention that the data that were considered as noise were: 398.639. Figure 12 shows the number of records in each table of the system.

7.2 Analysis using Google Analytics

In this subsection we analyse the effects of the system on athensvoice.gr using Google Analytics. We start by comparing a 1 month period of the use of the system with the same month of the previous year. We do this analysis for both distinct categories of visitors based on the device used. In order for our analysis to be complete we also deactivated the system for 2 days and we compare the data provided by Google's service against those of the previous week, while the system was being used.

Google Analytics is a service provided by Google for websites and applications, for a complete tracking of traffic. It can be used for the following:

- observation and documentation of traffic of a website for a specified period of time
- presentation of a number of statistical data for traffic, such as demographics

System DB Table	Records
initdata	27.718.535
user_from_to	10.393.173
tid_profile	13.693
user_has_read	19.188.534
user_profile	1.562.134
user_from_nid_to_nid	401.885
nid_profile	126.740
cosine_sim	53.406.449
day_tid	5.909.469
time_tid	6.494.281
day_profile	1.910
time_profile	1.463
noise	398.639
Total	**125.616.905**

Figure 12: Data Statistics of the System

- unique users of website
- total page views, and average page views per user
- average time a user spends on the website
- new and returning users
- technologies used by visitors in terms of operating system, browser, device, etc.
- referrals and other means the users access web pages from
- real time tracking of a website traffic

As can be seen by the above data, the service used for the analysis provides more than enough, in order for it to be thorough.

7.2.1 System in Use Analysis

In this section, we present the effects of using the system developed on athensvoice.gr. We do this by comparing a 1 month period (24/07/2015-24/08/2015) to the same period of the previous year (24/07/2014-24/08/2014). We perform this comparison to try and stay as much unaffected as possible by the change of users behaviour, for which the season and day plays an important factor. All of the comparisons are performed by separating the users into 2 groups, those that access Athens Voice using a desktop or tablet, and those using a mobile device. This is done because the web page differs for each device.

We start by showing the results of the use of the recommendation system, for desktop and tablet users. Figure 13 shows an overview chart of the increase of pages/session for the visitors of Athens Voice. It should be noted that this increase is despite the 9.58% decline of the total users of the website for this specific period of time.

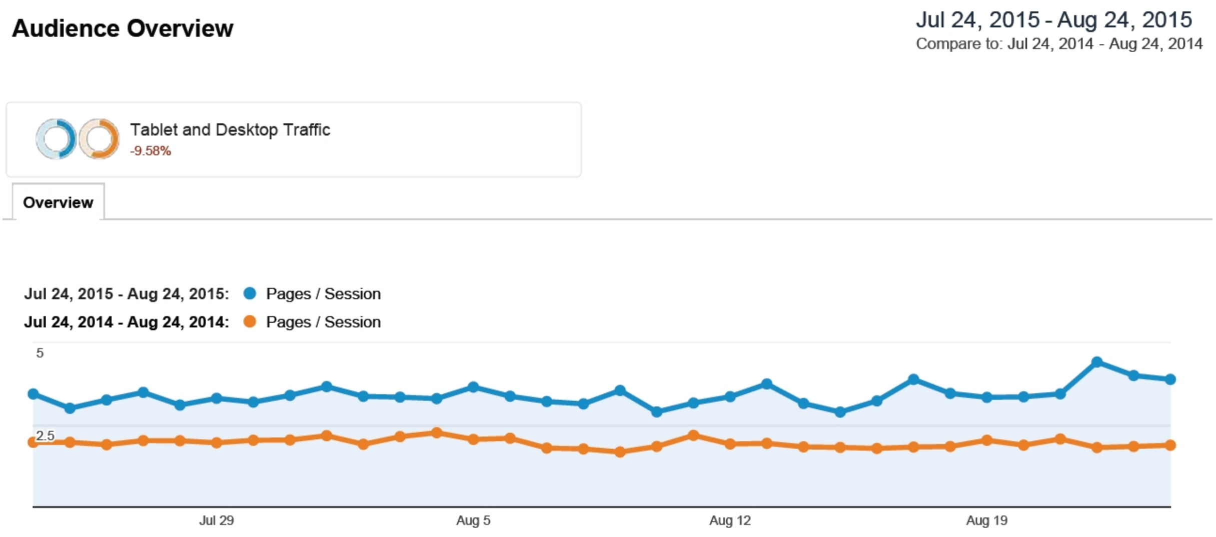


Figure 13: Chart for Desktop/Tablet visitors

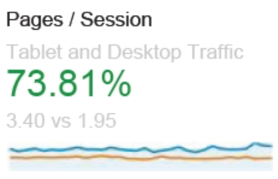


Figure 14: Percentage Increase for Pages/Session (Desktop/Tablet visitors)

Figure 14 shows the same increase of pages/session as a percentage, for this group of visitors. As can be seen by the figure, we have an approximate of 74% increase of the average pages viewed by visitors.

The following 2 figures, give us some insight as to how the same metric was affected for mobile device visitors of the website. Again we can see an increase in pages/session by this group of visitors, which although is smaller than the one of desktop and tablet users. It should be noted at this point that the total amount of traffic by mobile device users for this period of time was higher than the one of the previous year.

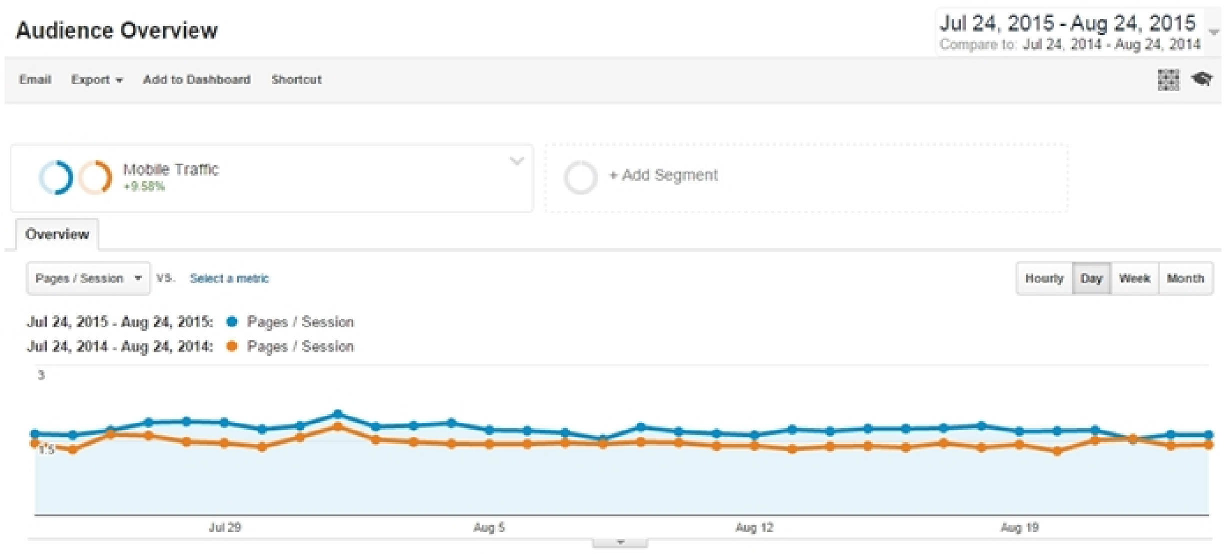


Figure 15: Chart for Mobile Devices visitors

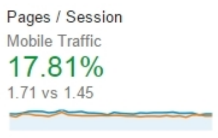


Figure 16: Percentage Increase for Pages/Session (Mobile Devices visitors)

7.2.2 System Deactivation Analysis

In order for the analysis of results to be as unaffected as possible by the changes in time and the general increase of internet users, we deactivated the system for 2 days, and made the same comparisons using Google Analytics for the same days of the previous week. The above scenario was executed for 29/09/2015-30/09-2015 and is compared with the data of 22/09/2015-23/09/2015.

Figure 17 shows an overview of the change in pages/sessions for the users of athensvoice.gr. It compares the data it has collected for the 2 days of the deactivation as opposed to those the system was being used. As can be seen by the figure, this metric has decreased during the period the system was deactivated, whereas the number of desktop and tablet users has increased on the website, and the respective one for mobile device users has dropped.

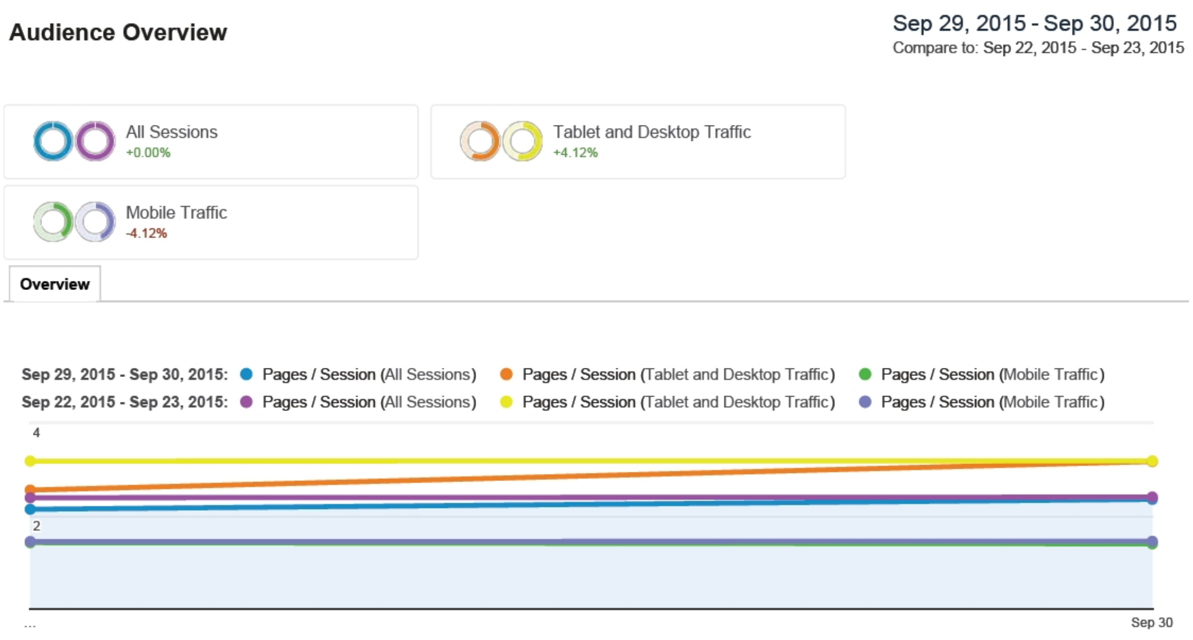


Figure 17: Comparison chart deactivated/activated System

Figure 18 shows the same metric in terms of percentage, as it was documented by Google's service. In this figure the decrease in pages per session is visualized in a more comprehensive way. The website of athensvoice.gr experienced an overall 6.31% drop which can be further analysed as a decrease of almost 11% for desktop and tablet users and 2.85% for mobile device users.

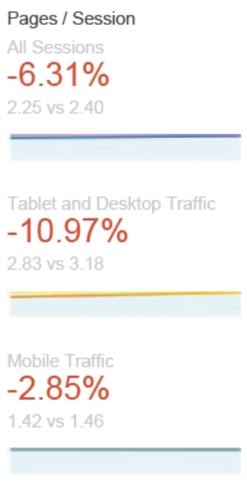
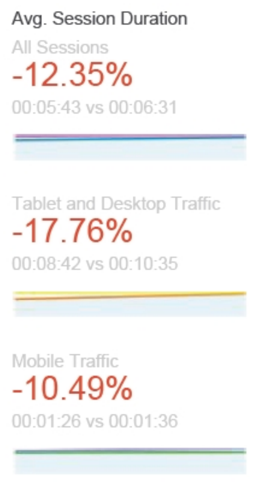


Figure 18: Percentage Increase for Pages/Session with System activated

From the above 2 figures the effects of the system on the website can be easily seen. It offers an overall increase of 6.3% in terms of pages per user, which in turn affect the total page views of the website, as well as the average time spent on the website for each visitor (as depicted in figure 19)


Avg. Session Duration

All Sessions
-12.35%
00:05:43 vs 00:06:31

Tablet and Desktop Traffic
-17.76%
00:08:42 vs 00:10:35

Mobile Traffic
-10.49%
00:01:26 vs 00:01:36

Figure 19: Average time spent on website during deactivation of system

7.3 Custom Analysis

In this section we analyse the data collected by the metric we developed. Similarly with the Google's analysis, we divide the results presented by taking into account the device used by the visitor of Athens Voice. The data collected and analysed, are of a 12 day period (10/09/2015-22/09/2015).

First we will analyse the results, as documented by the custom metric developed, in terms of absolute numbers. In figure 20 we show the performance of each of the criteria embedded into our system (y-axis), over the number of clicks they have produced (x-axis). As seen on the respective figure, desktop and tablet users prefer as criteria, that of the category profile one. The number of clicks produced by this specific criteria, is almost double of the ones of the next (which is based on the most read articles of the day). A very important factor to take into account at this point is that the number of recommendation served by each criteria differs, as described in previous sections. That being said, the ”most read” criteria, statistically is the one that serves the least number of recommendations (2). Despite that fact, desktop and tablet users, prefer it over the rest 3.

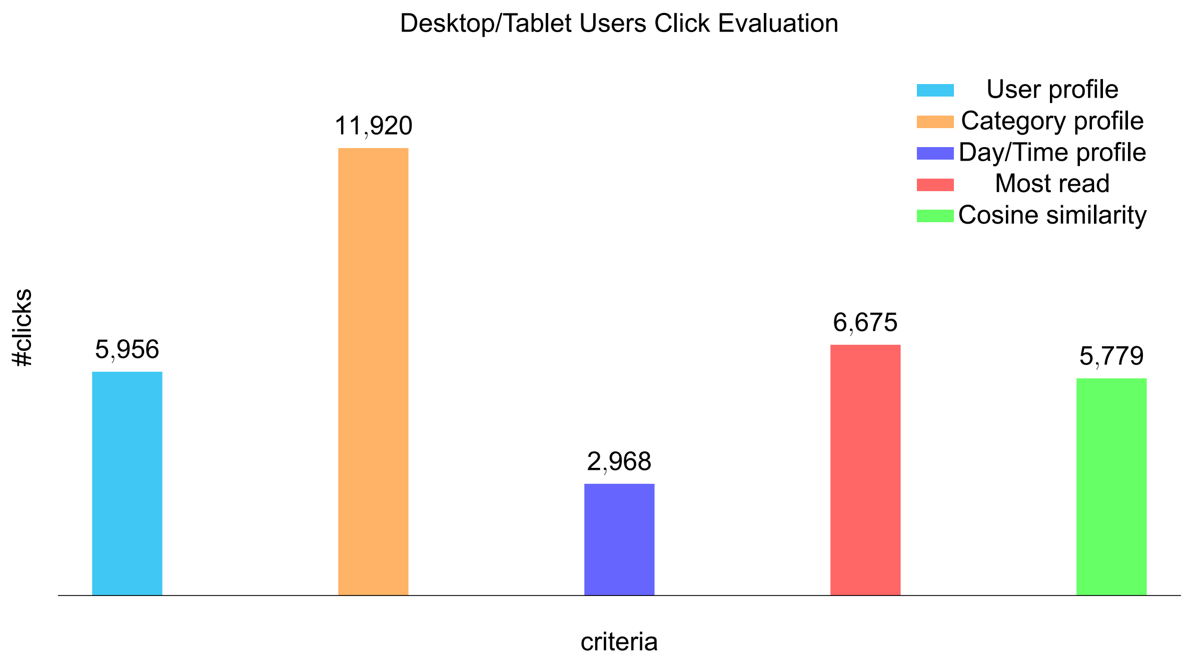


Figure 20: Click criteria choice for desktop/tablet users

Figure 21, shows the same analysis, for mobile device users. As we can see, the results differ than those documented for the other group of users. For mobile device visitors, the user profile criteria is the one that attracts the highest number of clicks, with the category one coming second. The rest 3 criteria are pretty close to each other, having a maximum of about 1,700 clicks as difference.

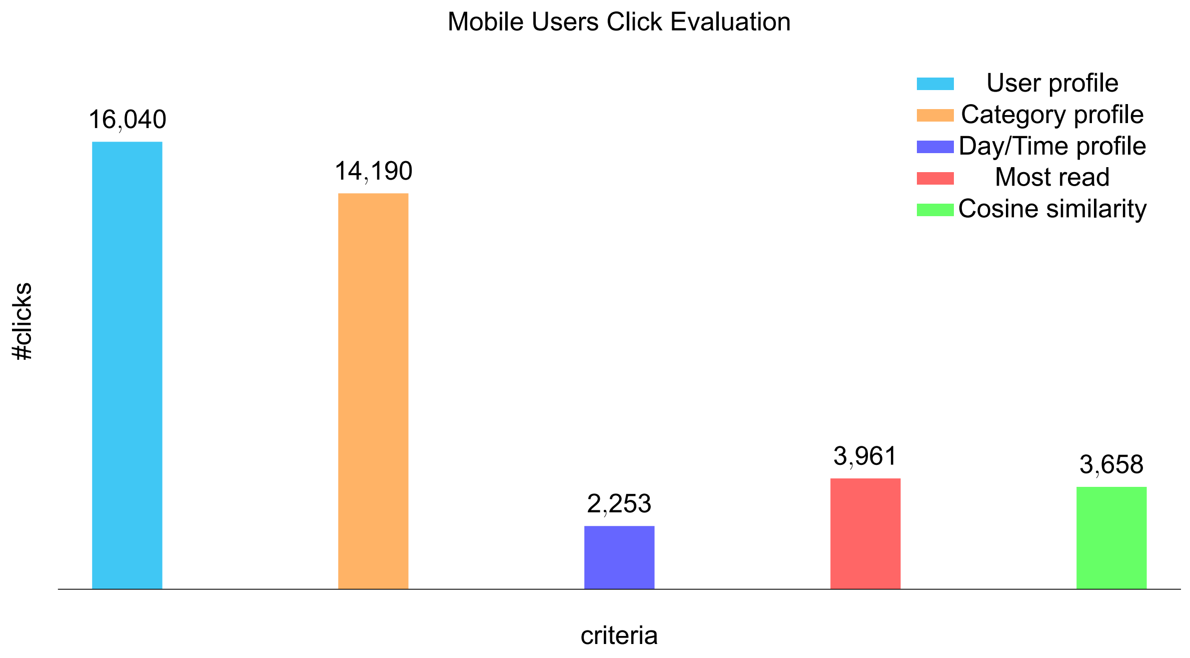


Figure 21: Click criteria choice for mobile device users

The above 2 figures, show the way criteria clicks are distributed, for the 2 groups of users of athensvoice.gr. A total of 74,030 number of clicks were documented for the specified period of time, out of which an approximate of 40,000 are those of the mobile device visitors and the rest for the desktop and tablet ones. With the exception of the category profile criteria, the rest show a fluctuation as far as their absolute numbers are concerned. However this can be easily explained, due to the common behaviour differences between the 2 groups of users.

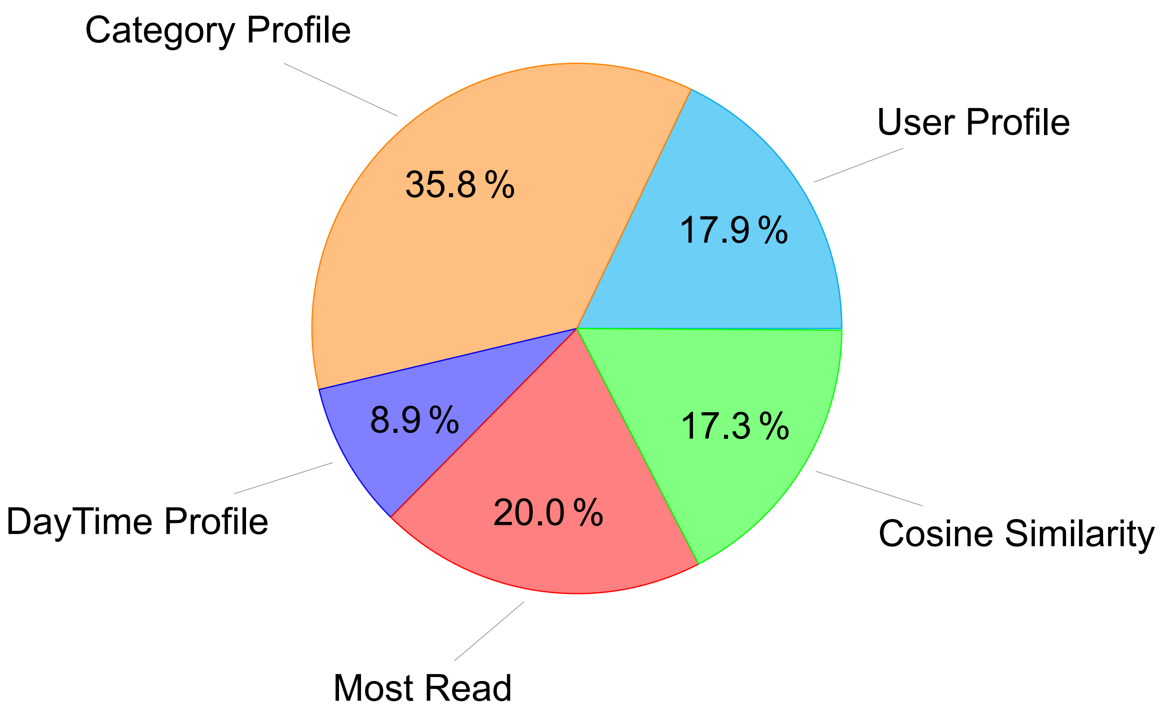


Figure 22: Percentages of click criteria by athensvoice.gr desktop/tablet users

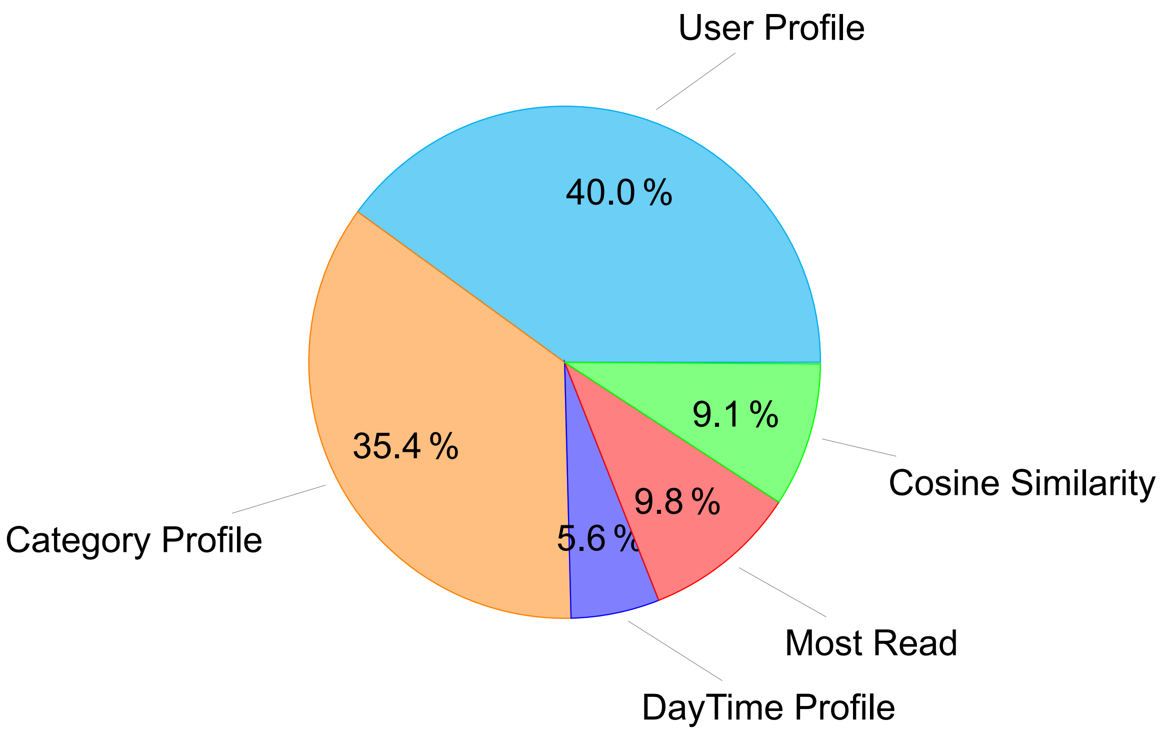


Figure 23: Percentages of click criteria by athensvoice.gr mobile devices users

Figure 22 and 23, show the same data in terms of percentage, whereas, figure 24, shows what the percentage of the criteria clicks documented by our assessment metric is throughout devices.

In figure 24 we present the overall performance of each of the criteria deployed into our recommendations system. This is done regardless of the device each visitor is using. We use this figure in order to show the importance of each criteria in our system in terms of the percentage over the total number of clicks collected. As can be seen by figure 24, the criteria that has given the system the most clicks, is that of the association between categories of the website. With a minor difference in terms of absolute numbers (a little above 5,000 clicks) the user profile criteria is chosen very frequently by athensvoice.gr visitors. The most read criteria and the one that takes into account the cosine similarity of articles, are close to each other (10,636 and 9,437 number of clicks respectively), and the criteria less frequently chosen by visitors of Athens Voice, is that of day and time.

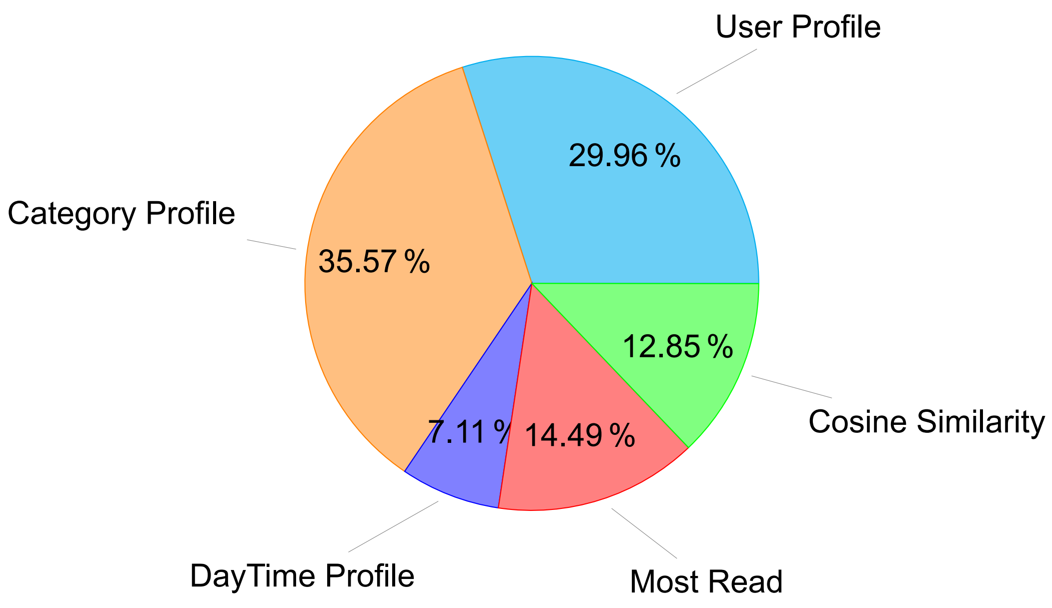


Figure 24: Accumulated percentages of click criteria by athensvoice.gr visitors

Another interesting factor to take into account, is the way the recommendations of the system are being clicked, based on the time of day. Figure 25 shows this data, and as can be seen, the time of day, greatly affects the clicks on recommendations, which is increased as the day progresses. As we can see by this figure, the recommendation clicks start off low during 02:00-07:59, which is logical if we take into account the decrease in the total number of visitors at that time. During the day 08:00-20:59, we experience an increase of almost 3 times as compared to the numbers of the previous time period. Finally we have the 21:00-01:59 time period of each day, in which the highest number of clicks are been shown.

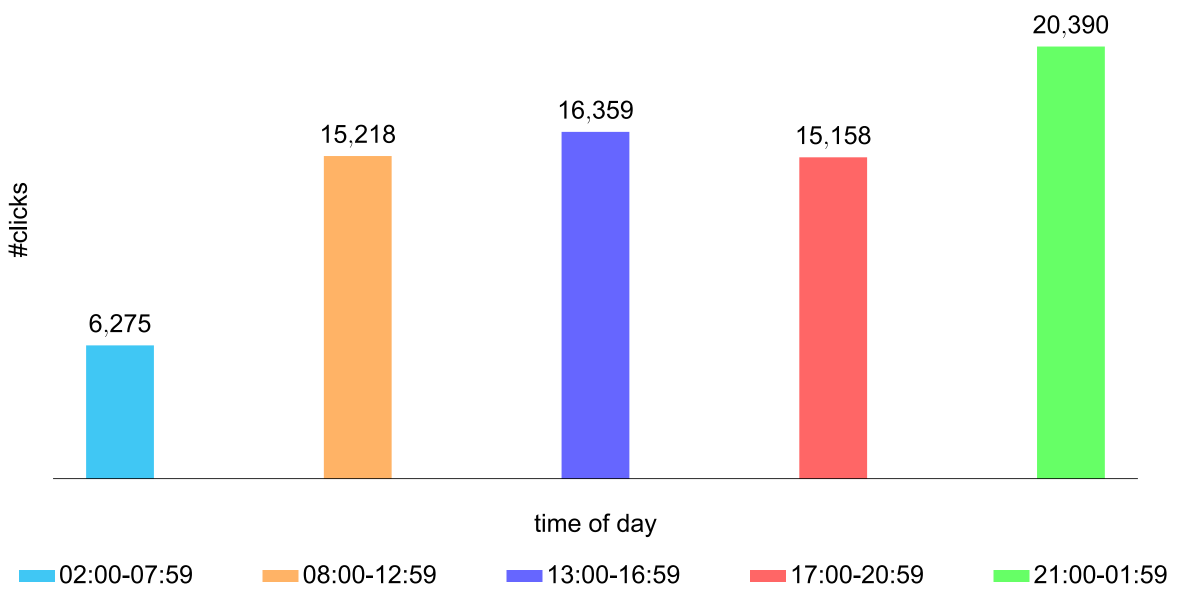


Figure 25: Clicks on recommendations based on time of day

7.4 Benefits of Integration

In the previous sections we analysed the results of our system's integration within the website of athensvoice.gr. In this section, we will present the benefits of its integration in terms of both absolute numbers as well as economic values.

In order to make this analysis we will investigate 2 scenarios, best and worst case ones. The best case scenario is the one in which all of the clicks made to recommendations produced by our system would not happen if our system was not live. The worst case scenario we will investigate is that only 25% of those clicks occurred due to the integration of our recommendation system.

Prior to our analysis, we should calculate the number of impressions produced in the best case scenario on a one month period. As we have shown in the previous section, during the 12 day period, a total of 74,000 impressions were documented. Assuming that the same behaviour was documented for the rest 18 days of the month, we have a total of 185,000 new page views.

Before moving on with our analysis, it should be noted that at the website of athensvoice.gr there are 6 slots for advertisement banners into each page (1 skin, 1 728x90 banner, 3 300x250 banners, 1 text link). These slots, if sold by the advertisement department of Athens Voice, have an average CPM (cost per million impressions) of 1.5€, based on the web page position they are shown.

Revenue (/month)	**Worst Case (46,250 impr.)**	**Best Case (185,000 impr.)**
From 1 banner	69.38€	277.5€
Total from 6 banners	416.3€	1,665€

Figure 26: Economic benefit of integration in best/worst case scenario

The table in figure 26 depicts the economic benefits of our system's integration into the website of Athens Voice. Based on the above economic values we can calculate the yearly turnover of our recommendation system as follows:

- 4,994€, in the worst case scenario.
- 19,980€, in the best case one.

It should be noted at this point that the above figures are proportional to the following 4 factors:

1. Athens Voice's website monthly traffic, should our system be integrated into a website with higher or lower monthly traffic, these economic values will change accordingly.
2. Number of recommendations served in total and design. These economic values were calculated given the fact that a total of 15 recommendations were served. Should the total number of recommendation increase or decrease the economic benefits will change accordingly. Moreover, since the design of a web page plays an important role in a user's behaviour, we believe that if the recommendations served were changed in design these numbers will also change.
3. Placement of our system's recommendation on the web page. We believe that the slot in which our recommendations are shown plays an important part to them being viewed and clicked by visitors. Should this placement change, the economic values presented on this section will also change.
4. Total number of advertisement slots on web page and average CPM. The above described economic values were calculated for the website of athensvoice.gr. Shoudl our system be integrated into a website that has a higher or lower number of advertisement slots or these slots are sold at a different CPM on average, these numbers will change accordingly.

8. CONCLUSION AND FUTURE WORK

8.1 Conclusion

In this work we developed a hybrid approach to recommendation systems, in which multiple criteria was used and collaborative and content-based filtering techniques were combined. The novelty of this approach is due to the way the combination of techniques is done, through the 5 criteria chosen as the base of the recommendation algorithm, its real-time deployment on a live website and the custom metric developed.

This work was deployed on a live website, that of athensvoice.gr, and served its recommendations real-time. As a result of this, the time allowed for the recommendations to be calculated and served back to the visitor, was limited, and special care was taken in order for it to be as reduced as possible. On the upside, the training, run and assessment of the recommendation system was done using real-time data.

Moreover, a custom metric was introduced, in which the tracking of the criteria by which the recommendation chosen by each user was performed. In this way, we gave the system developed a self-assessment method, and to us a useful tool, to perform our analysis over the results.

The use of our recommendation system on the website of athensvoice.gr, lead to an average increase of 6.3% in the number of web pages viewed by a visitor, or in terms of numbers, to a total of almost 74,000 clicks(page views), in 12 days.

8.2 Future Work

In the future, we plan on expanding our recommendation system in various ways, as depicted below:

- by embedding social media features, we intend on accessing demographic characteristics of each visitor, data that can be used both for a new criteria in our algorithm, as well as a new analysis method for our results.
- experiment with the acceptance by visitors of articles already read into the recommendations served.
- by using the location of the user, along with the one the article is about, we would like to explore the results of applying such a criteria in our recommendation algorithm.
- we would also like to experiment with different spots on the web-page for the recommendations to be shown, in order to find the one that serves the best results, and possibly leading to a custom page for each visitor of the website.
- finally, by using our custom metric tool, we plan on integrating into our system, machine learning approaches, that use the data collected by the tool developed, in order to adapt the number and order of the recommendations served by each criteria.

A. TRACKER SCHEMA TABLE DESCRIPTION

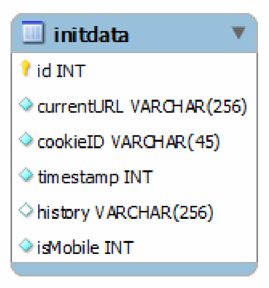


Figure 27: Table initdata.

Table initdata has the following fields:

- id, primary key for the table
- currentURL, current url visited
- cookieID, unique id given by the system for the visitor
- timestamp, timestamp of the visit
- history, previous url visited
- isMobile, 1 or 0 if visitor is using a mobile device or not

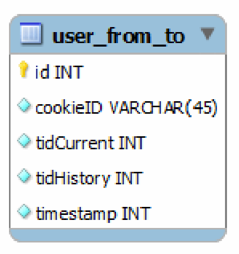
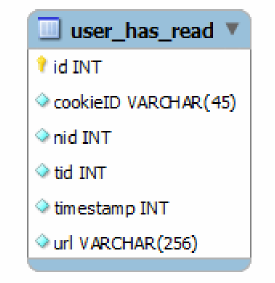


Figure 28: Table user_from_to.

Table user_from_to has the following fields:

- id, primary key for the table
- cookieID, unique id given by the system for the visitor
- tidCurrrent, current category id
- tidHistory, previous category id
- timestamp, timestamp of the click

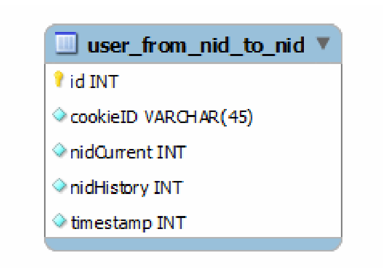


user_has_read
id INT
cookieID VARCHAR(45)
nid INT
tid INT
timestamp INT
url VARCHAR(256)

Figure 29: Table user_has_read.

Table user_has_read has the following fields:

- id, primary key for the table
- cookieID, unique id given by the system for the visitor
- nid, article id
- tid, category id
- timestamp, timestamp of the read
- url, url of the article

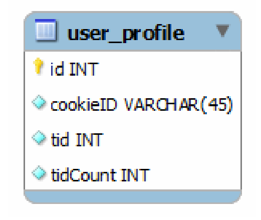


user_from_nid_to_nid
id INT
cookieID VARCHAR(45)
nidCurrent INT
nidHistory INT
timestamp INT

Figure 30: Table user_from_nid_to_nid.

Table user_from_nid_to_nid has the following fields:

- id, primary key for the table
- cookieID, unique id given by the system for the visitor
- nidCurrrent, current article id
- nidHistory, previous article id
- timestamp, timestamp of the click



user_profile
id INT
cookieID VARCHAR(45)
tid INT
tidCount INT

Figure 31: Table user_profile.

Table user_profile has the following fields:

- id, primary key for the table
- cookieID, unique id given by the system for the visitor
- tid, category id
- tidCount, number of times read

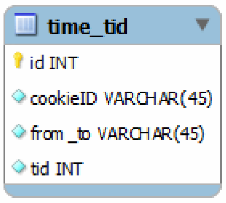


Figure 32: Table time_tid.

Table time_tid has the following fields:

- id, primary key for the table
- cookieID, unique id given by the system for the visitor
- from_to, timezone in which the read happened. One of the values:
  - 02:00-07:59
  - 08:00-12:59
  - 13:00-16:59
  - 17:00-20:59
  - 21:00-01:59
- tid, category id

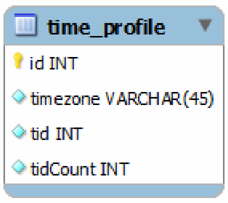


Figure 33: Table time_profile.

Table time_profile has the following fields:

- id, primary key for the table
- timezone, timezone in which the read happened
- tid, category id

- tidCount, number of times read

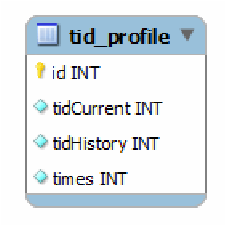


Figure 34: Table tid_profile.

Table tid_profile has the following fields:

- id, primary key for the table
- tidCurrent, the current category read
- tidHistory, previous category
- times, number of times current was clicked right after history

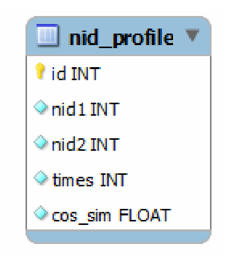


Figure 35: Table nid_profile.

Table nid_profile has the following fields:

- id, primary key for the table
- nid1, id of the first article
- nid2, id of the second article
- times, number of times nid1 and nid2 were clicked successively
- cos_sim, cosine similarity between the 2

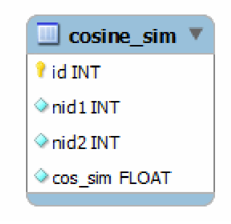


Figure 36: Table cosine_sim.

Table cosine_sim has the following fields:

- id, primary key for the table
- nid1, id of the first article
- nid2, id of the second article
- cos_sim, cosine similarity between the 2

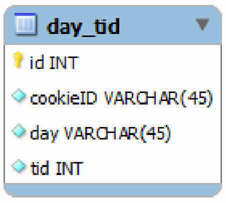


Figure 37: Table day_tid.

Table day_tid has the following fields:

- id, primary key for the table
- cookieID, id of the visitor
- day, day of the week
- tid, category id read

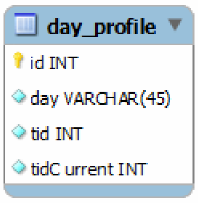


Figure 38: Table day_profile.

Table day_tid has the following fields:

- id, primary key for the table
- day, day of the week
- tid, category id read
- tidCount, number of times read

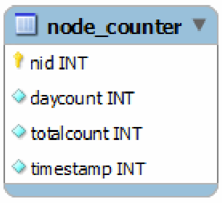


Figure 39: Table node_counter.

Table node_counter has the following fields:

- nid, id of the article
- totalcount, total number of reads
- daycount, today’s reads
- timestamp, timestamp of last read

B. CLICK TABLE DESCRIPTION

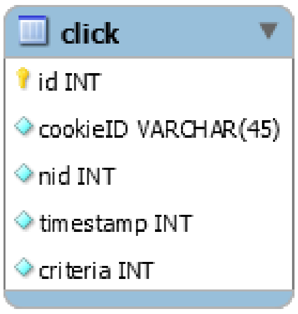


Figure 40: Table click.

Table click, has the following fields:

- id, primary key for the table
- cookieID, unique id given by the system for the visitor
- nid, id of the article selected by the visitor
- timestamp, timestamp of the click
- criteria, criteria by which the recommendation was produced. One of the following:
  - 1, if the recommendation was produced using the user's profile
  - 2, if the recommendation was produced by the category's association with another
  - 3, if the specific day and time was used
  - 4, if the article is one of the most read of the day
  - 5, if cosine similarity was used in order to recommend it

REFERENCES

[1] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In Proceedings of the 16th international conference on World Wide Web (WWW '07). ACM, New York, NY, USA, 271-280.

[2] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. on Knowl. and Data Eng. 17, 6 (June 2005), 734-749.

[3] R. Burke, Hybrid Web Recommender Systems: Survey and Experiments, User Modeling and User-Adapted Interaction, Volume 12, Issue 4, 2002, p. 331–370.

[4] Florent Garcin, Christos Dimitrakakis, and Boi Faltings. 2013. Personalized news recommendation with context trees. In Proceedings of the 7th ACM conference on Recommender systems (RecSys '13). ACM, New York, NY, USA, 105-112.

[5] Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. 2014. Offline and online evaluation of news recommender systems at swissinfo.ch. In Proceedings of the 8th ACM Conference on Recommender systems (RecSys '14). ACM, New York, NY, USA, 169-176.

[6] G. Gebremeskel and A. P. de Vries, The Degree of Randomness in a Live Recommender Systems Evaluation, in Working Notes of CLEF 2015 – Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015., 2015.

[7] Mustansar Ali Ghazanfar and Adam Prugel-Bennett. 2014. Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems. Expert Syst. Appl. 41, 7 (June 2014), 3261-3275.

[8] Jure Leskovec, Anand Rajaraman, Jeff Ullman. Mining of Massive Datasets. Cambridge University Press. California, 2014.

[9] Jiahui Liu, Peter Dolan, and Elin Ronby Pedersen. 2010. Personalized news recommendation based on click behavior. In Proceedings of the 15th international conference on Intelligent user interfaces (IUI '10). ACM, New York, NY, USA, 31-40.

[10] Zhongqi Lu and Zhicheng Dou and Jianxun Lian and Xing Xie and Qiang Yang. 2015 . Content-Based Collaborative Filtering for News Topic Recommendation. AAAI Conference on Artificial Intelligence.

[11] Andrii Maksai, Florent Garcin, and Boi Faltings. 2015. Predicting Online Performance of News Recommender Systems Through Richer Evaluation Metrics. In Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15). ACM, New York, NY, USA, 179-186.

[12] Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2013. News vertical search: when and what to display to users. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval (SIGIR '13). ACM, New York, NY, USA, 253-262.

[13] Robert Palovics, Andras A. Benczur, Levente Kocsis, Tamas Kiss, and Erzsebet Frigo. 2014. Exploiting temporal influence in online recommendation. In Proceedings of the 8th ACM Conference on Recommender systems (RecSys '14). ACM, New York, NY, USA, 273-280.

[14] J. Ben Schafer, Joseph A. Konstan, John Riedl. E-Commerce Recommendation Applications. GroupLens Research Project, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455.

[15] Diego Saez-Trumper, Daniele Quercia, and Jon Crowcroft. 2012. Ads and the city: considering geographic distance goes a long way. In Proceedings of the sixth ACM conference on Recommender systems (RecSys '12). ACM, New York, NY, USA, 187-194.

[16] Xiaolan Sha, Daniele Quercia, Pietro Michiardi, and Matteo Dell'Amico. 2012. Spotting trends: the wisdom of the few. In Proceedings of the sixth ACM conference on Recommender systems (RecSys '12). ACM, New York, NY, USA, 51-58.

[17] Jeong-Woo Son, A-Yeong Kim, and Seong-Bae Park. 2013. A location-based news article recommen-

dation with explicit localized semantic analysis. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval (SIGIR ’13). ACM, New York, NY, USA, 293-302.

[18] Liang Tang, Yexi Jiang, Lei Li, Chunqiu Zeng, and Tao Li. 2015. Personalized Recommendation via Parameter-Free Contextual Bandits. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’15). ACM, New York, NY, USA, 323-332.

[19] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR ’06). ACM, New York, NY, USA, 501-508.

[20] Saul Vargas and Pablo Castells. 2011. Rank and relevance in novelty and diversity metrics for recommender systems. In Proceedings of the fifth ACM conference on Recommender systems (RecSys ’11). ACM, New York, NY, USA, 109-116.

[21] Jialei Wang, Steven C.H. Hoi, Peilin Zhao, and Zhi-Yong Liu. 2013. Online multi-task collaborative filtering for on-the-fly recommender systems. In Proceedings of the 7th ACM conference on Recommender systems (RecSys ’13). ACM, New York, NY, USA, 237-244.

[22] Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. 2012. BlurMe: inferring and obfuscating user gender based on ratings. In Proceedings of the sixth ACM conference on Recommender systems (RecSys ’12). ACM, New York, NY, USA, 195-202.

[23] Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. 2013. Personalized next-song recommendation in online karaokes. In Proceedings of the 7th ACM conference on Recommender systems (RecSys ’13). ACM, New York, NY, USA, 137-140.

[24] S. Werner and A. Lommatzsch, Optimizing and Evaluating Stream-based News Recommendation Algorithms, in Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014, 2014, pp. 813-824.

[25] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond clicks: dwell time for personalization. In Proceedings of the 8th ACM Conference on Recommender systems (RecSys ’14). ACM, New York, NY, USA, 113-120.

[26] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. 2014. Addressing cold start in recommender systems: a semi-supervised co-training algorithm. In Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval (SIGIR ’14). ACM, New York, NY, USA, 73-82.

[27] Weinan Zhang, Jun Wang, Bowei Chen, and Xiaoxue Zhao. 2013. To personalize or not: a risk management perspective. In Proceedings of the 7th ACM conference on Recommender systems (RecSys ’13). ACM, New York, NY, USA, 229-236.