Abstract: In this paper we present a mobile application based on a hybrid multi-objective genetic algorithm and Natural Language Processing that can be used by tourists to smartly generate itineraries. Besides allowing users to find interesting Points of Interest based on highly detailed information, the semantic search approach is also capable of providing recommendation without requiring a lot of information regarding the current user. A multi-objective genetic algorithm has been used in order to find Pareto-optimal solutions in near real-time for the itinerary problem. The proposed system requires minimum input from the user and it is easy to use, thus we can state that our solution is both complex and at the same time user-oriented.

Key-Words: semantic similarity, free text document indexing, multi-objective genetic algorithm, itinerary recommender system

1 Introduction
Nowadays, due to the developments that occurred in the mobile devices industry, there is a great interest in developing efficient applications. One of the problems which are of great interest for this domain, both from the commercial and research point of view, is related to the design of flexible, efficient, user-friendly mobile tour guide applications.

One of the first attempts in this field is represented by the Cyberguide project, which is a platform, in which the authors are developing prototypes for mobile applications that are targeted to model the tour guide activity. Furthermore, this project is characterized by context awareness meaning that it takes advantage of the information provided by the mobile device related to the context (GPS coordinates of the present location of the user, etc.). This real time information is used to offer adaptable and dynamic tour guide itinerary generation. The paper [1] introduces Cyberguide and presents the overall architecture and development procedures for multiple different hand-held platforms. The authors also discuss the general research issues that have emerged in
context-aware applications development in a mobile environment.

Another approach is presented in [2]. The main purpose of Guide system is to overcome the limitations of the traditional information and navigation tools available to city visitors. The solution is applied to Lancaster city. This system uses mobile computing and wireless infrastructure and recommends points of interest based on the user’s profile and on the context. The business flow was developed by interacting with professional tourist guides. Paper [2], presents also an evaluation of the system by focusing on the quality of the visitor’s experience with the system. However, the drawback of this system was related to the effort required to record preferences so that to customize a tour. Guide offers users the possibility of creating a tour by selecting objectives, presenting their domains of interest and preferences. The restrictions that the system takes into account refer to the schedule of objectives, the best visiting intervals, etc. The tour can be changed dynamically, based on the actual time the tourist already spent at some attractions or on weather conditions. The user profile is also continuously updated, using information from previous trips.

[3] presents an artificial intelligence and metaheuristic approach for dealing with the tourist planning issue on mobile devices. The solution presented offers a decision support and a modeling mechanism starting from the orienteering problem. The problem involves a set of candidate objectives for which a score is being associated. The problem is thus reduced to maximizing the total score of the visited places, while complying with the constraints related to the total amount of time or the total distance. The score of a Point of Interest - POI is defined as the interest of a tourist in that place. Scores are calculated using the vector space model. The trip planning problem is solved using a guided local search metaheuristic. In order to compare the performance of this approach with an algorithm that appeared in the literature, both are applied to a real data set from the city of Ghent.

Our approach is also based on semantic search as [3] but uses neuronal network based document indexing mechanisms. Furthermore, we present the user a list of possible points of interest as [2], but we dynamically create the tour and we don’t require data introduction effort from the user. Moreover, our approach also generates public transport routes to link objectives and also offers information regarding the pollution level from the areas in which pollution sensors are available. The trip planner presented in this paper is tested in Bucharest city.

This paper is organized as follows: in the second section we formalize the model used for generating itineraries, the third part we present the overall distributed architecture of our web service based platform, while in the fourth section we enlarge on the scenario and introduce the mobile itinerary recommender application. The last section will conclude the article and present the future research guidelines.

2 Evolutionary Model for Itinerary Generation in Mobile Applications

In order to offer users the possibility of flexibly selecting the itinerary based on their preferences we proposed an algorithm for indexing documents and a flexible semantic search mechanism.

2.1 Document indexing for extracting candidate POIs

In order to determine the possible candidate points of interest, we developed a semantic search algorithm that has two main steps: the semantic classification of the description files of POI and also a document indexing step. Afterwards we perform the semantic matching between the indexed words and the search phrases introduced by the user.

Document indexing is a useful in a lot of Natural Language Processing application such as text mining, knowledge retrieval, keywords extraction, etc [4]. Indexing defines the process of converting a document into a list of words included in it. The result of indexing is represented by a list of words called index language [5].

The selection process of document indexing has the following steps:

Step 1: Eliminate stop words. Stop words are words which have grammatical functionality, but are not related to the content of the document. Some possible stop words are: conjunction, prepositions, auxiliary verbs, articles, pronouns, etc.

Step 2: For the remaining of the document words in the list perform word stemming. This means that we extract the root of the words. In order to achieve this, we use Porter stemmer for English language.

Step 3: This step consists in performing a neuronal network based filtering so that to extract only those words that are relevant to the context.

The input nodes for the neural network are represented by the following features that can be grouped into statistical features (TF, IDF, ITF) [5], position features (T, F_{pos}) [6], grammatical syntax (subject, object, predicate (SPO)):
term frequency (TF) - the frequency of each term in the document;

- inverted document frequency (IDF) - the number of documents that contain the word from the initial document sample;

\[ \text{IDF} = \log_2 \frac{\text{TF}}{\text{Tot}} \text{ , where Tot - total occurrence} \] [7]

- inverted total frequency (ITF) - total frequency of the word in the initial document sample;

- T - Boolean value indicating whether the word appears in the title or not;

\[ F_{pos} = \frac{1}{\sqrt{i}} \text{ , where } i \text{- the number of the sentence in which the word appears for the first time} \]

- SPO - Boolean value indicating whether the word is a subject, predicate or an object in the sentences from the first paragraph. For determining the grammatical category of a word we used Gate and we integrated the Stanford Parser [8][9].

Therefore, the neuronal network will have six input nodes. Other characteristics of the ANN are: one hidden layer, usually it is not indicated to use more than one hidden layer; the number of hidden nodes is given by the formula:

\[ N_h = \frac{\text{no. attributes} + \text{no. classes}}{2} \]

where:

- no. attributes - represents the number of features (in our case equals six)

- no. classes - represents the number of classes in which words can be classified (in our case 2: keyword and non-keyword).

The artificial network model can be seen in Figure 1.

![Artificial Neuronal Network](image)

Fig. 1 Artificial neuronal network for extracting relevant words

In order to train the neuronal network we used the back propagation algorithm implemented in Weka [10] on a set of manually classified instances. For the training set we have chosen documents from different sources, with a focus on newspaper articles. We created a java application that automatically extracts words in the document by following the steps presented above and computes the values for them. We then manually classify the extracted potential keywords into two groups: keywords and non-keywords. We use the resulting set of classified instances to train the ANN.

We tested the artificial neuronal network on a set of 50 English documents. From the set of 3850 identified possible keywords, only 10% were selected in the manual classification phase. In order to choose the best configuration for the ANN, several choices for the number of neurons in the hidden layer have been tested.

Below, we present the comparison results for 3 ANN configurations:

- \( N_h = (\text{no. attributes} + \text{no. classes})^2 \) with the results:
  
  Confusion Matrix = \[
  \begin{pmatrix}
  43 & 342 \\
  29 & 3436
  \end{pmatrix}
  \]

  - Correctly Classified Instances = 3479 90.3636 %
  - Incorrectly Classified Instances = 371 9.6364 %

- \( N_h = \text{no. attributes} + \text{no. classes} \), with the results:

  Confusion Matrix = \[
  \begin{pmatrix}
  44 & 341 \\
  45 & 3420
  \end{pmatrix}
  \]

  - Correctly Classified Instances = 3464 89.974 %
  - Incorrectly Classified Instances = 386 10.026 %

- \( N_h = 10 \), with the results:

  Confusion Matrix = \[
  \begin{pmatrix}
  45 & 340 \\
  49 & 3416
  \end{pmatrix}
  \]

  - Correctly Classified Instances = 3461 89.8961 %
  - Incorrectly Classified Instances = 389 10.1039 %

- We also compared the results with the Naive Bayesian Classifier:

  Confusion Matrix = \[
  \begin{pmatrix}
  35 & 350 \\
  28 & 3437
  \end{pmatrix}
  \]

  - Correctly Classified Instances = 3472 90.1818 %
  - Incorrectly Classified Instances = 378 9.8182 %

We conclude that the ANN configuration initially presented gives us better results. After the document indexing phase is completed, we can perform the semantic matching.

First in order to understand the sense of the words as it is used in the documents we perform
word disambiguation. For this, we query an external corpus, YAGO [11], based on Wikipedia and WordNet that contains both word definitions and the semantic relations with other words in the same synonymic class. For word sense disambiguation we identify all possible senses of the words that were previously POS – Parts of Speech tagged. For each meaning of the word we identify all possible solutions:

- Its own definition that includes example texts that WordNet or other corpus provides to the glosses.
- The dictionary definition of the synonymic groups that are connected to it through the “has a” relations. If there is more than one relation for a word sense, then the glosses for each relation are concatenated into a single gloss string.
- The dictionary definition of the synonymic group that are connected to it through the “is a” relations.
- The dictionary definition of the synonymic group that are connected to it through the “part of” relations.

After these steps are taken, the semantic similarity of the synonymic group the words belong to is measured. If a word has more than one sense, it will appear in multiple synonymic groups at various locations in the classification. WordNet defines relations between synonymic groups and relations between word senses. To measure the semantic similarity between two synonymic groups, the most appropriate relations that are to be used are the semantic hierarchic ones. Several types of semantic similarity formulas exist: Conrath, Lin, Resnik. In our approach, Lin formula was chosen as it fully takes into account the information content: 

\[ \text{Sim}_{\text{Lin}}(X_1, X_2) = \frac{2 \cdot S(X_1, X_2)}{\text{IC}(X_1) + \text{IC}(X_2)}. \]

where:

\[ \text{Sim}_{\text{Lin}}(X_1, X_2) \] – the level of similarity between concepts 1 and 2 computed by Lin formula

\[ S(X_1, X_2) \] - the degree of shared information for \( X_1 \) and \( X_2 \) concepts

\[ X_1, X_2 \] - concepts 1 and 2

\[ \text{IC}(X) \] - information content (IC) of X concept.

The semantic matching evaluation will be performed against the user key search phrases and the keywords extracted from the documents.

For instance, if a tourist introduces “modern art” as search phrase, and in a document about a military museum it is specified that there is a painting created by a modern art painter “Paul Gauguin”, then the POI will be suggested to the user.

2.1 Multi-objective evolutionary algorithm for generating itineraries

Besides the semantic search approach described in the previous section, users can also choose POI from a suggestion list built using collaborative filtering. All selected POIs are displayed in a common list, allowing the user to change their importance, by changing the order.

In order to generate itineraries, we use a hybrid genetic algorithm approach, as no exact algorithm that can solve the problem in real-time is known [12]. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. In order to decrease the number of required generations, we use the following heuristic: each time we add a new POI to an itinerary we choose the one with the highest ratio between the score, \( S_i \) and the time required to visit the POI.

We consider that the itinerary has a start time \( T_S \), a finish time \( T_F \) and must be generated from a set of \( n \) locations, each location having a score \( S_i \), an opening time \( O_i \), a closing time \( C_i \) and a visit duration \( D_i \). In order to increase the performance with compute in advance the minimum \( t_{ij}^{\text{min}} \) and maximum time \( t_{ij}^{\text{max}} \) required to travel between locations \( i \) and \( j \). The values can be calculated for different hours or week days. The algorithm is run until a maximum number of generations, \( G_{MG} \) is reached or until the best found solution doesn’t change for \( G_{CG} \) consecutive generations. Depending on the purpose \( G_{MG} \) can be adjusted either for accuracy or speed. We have chosen an Elitist approach [13] in which we use two populations, the normal one \( P_g \) and the population of non-dominated solutions in generation \( g < G_{MG} \), \( E_g \).

**Initialization:** For \( g = 0 \), we create the initial population \( P_0 \), with a number of \( N_p \) itineraries called chromosomes. \( E_0 = \emptyset \). In order to limit the number of required algorithm iterations, all itineraries in the initial population are generated valid. The first POI in each itinerary is chosen in a random manner. All the next POIs will be inserted only if the arrival time \( A_i < C_i - D_i \), where \( A_i = A_{i-1} + D_i \), \( i \in [1, n] \) and if \( A_i + D_i < T_F \).

**Evaluation:** We evaluate all \( N_p \) itineraries based on the following three criteria. For itinerary \( j \) we compute \( VT_j = \sum_{k=1}^{n_j} D_k \) - the useful visiting time,

\[ MR_j = \sum_{k=1}^{n_j} S_k \] - the medium ranking of the POIs in the itinerary,

\[ DV_j = n_j \] - diversity considered as the

\[ IC(X) - \text{information content (IC) of X concept}. \]
number of POIs in the itinerary. We evaluate all the itineraries in order to find the non-dominated ones.

**Selection:** As we have chosen an elitist approach in which we copy all non-dominated solutions from $P_g$ in $E_g$. We remove all other solutions from $P_g$ and we automatically include all solutions from $E_g$ in $P_{g+1}$. Using mutation and crossover we generate $N_p - |E_g|$ solutions that are added to $P_g$ in order to have $|P_{g+1}| = N_p$. If $|E_g| > N_p/2$ we use a clustering approach to select $N_p/2$ solutions as different as possible.

**Mutation:** Adds random variation to the evolving population. We randomly choose 2 links in the itinerary and we remove all the locations between. We then complete the itinerary by randomly inserting new POIs.

**Crossover:** Combines the features of two parent chromosomes to form new children by swapping corresponding itinerary segments of the parents. We first randomly select a link offset. We then swap the resulting segments by removing the POIs with the lowest $S_i$ if necessary in order to keep the itineraries valid.

3 **Itinerary recommender system**

The proposed solution presented in Figure 2 relies on a multi-tier paradigm for both the server and the client implementations. Even though a client only architecture would offer several benefits, such as the possibility to work entirely offline, it is not a feasible option for our approach due to the big amounts of data used both by the semantic search and by the collaborative filtering algorithms. Therefore, we have chosen a mixed approach in which the resource intensive computations are performed on the server, while the simple ones are performed directly on the client. All the main functions are implemented as semantically annotated web services so they can be used in automatic or semi-automatic web service composition frameworks like the one we presented in [14]. The Interface Layer supports communication with clients using simple http request, web service calls and socket connections in order to allow a maximum flexibility in regard to the technologies chosen to implement the client applications. The reference client implementation for shown in Figure 3 relies on the latest standard web technologies such as WebSQL [15] and WebStorage [16] to offer portability across different platforms as well as a rich user experience.

W3C GeoLocation API [17] was used to determine and monitor the position of the user.

![Fig. 3 Mobile HTML5 interface](image)

In order to use the application, the tourist can both semantically search for POI as described in this paper or can select suggested POIs. The suggestions are generated using a collaborative filtering algorithm that also takes into consideration the current context. The itinerary is generated by taking into account the traveler’s position, time, budget, weather and statistic information recorded in the system about the visit duration for any POI. In the case that the tourist exceeds the average duration for some of the points of interest, the itinerary is dynamically updated. Moreover, we implement a crowd sourcing approach in order to determine the real visiting duration for the users of the application. For each user, the system constantly updates the difference for each category of POI between the current user’s visiting duration and the average duration. This information is used to better predict the visiting durations in future itineraries. Public transport route finding is also included in the
application in order to allow the user to easily travel between the locations selected in the itinerary.

5 Conclusions
In this paper we presented an itinerary recommender system that uses both semantic search and collaborative filtering to allow users to find interesting Points of Interest. A multi-objective itinerary hybrid genetic algorithm that allows finding routes in near real-time was also presented. In the future we want to implement a crowdsourcing approach in order to compare the difference between the estimated and real times need to travel between POIS. The semantic search approach allows users to find POIs related to highly specific information.

6 Acknowledgments
Some parts of this article are results of the projects: 1. „Doctoral Program and PhD Students in the education research and innovation triangle”. This project is co-funded by European Social Fund through The Sectorial Operational Programme for Human Resources Development 2007-2013, coordinated by The Bucharest Academy of Economic Studies (Project no. 7832, “Doctoral Program and PhD Students in the education research and innovation triangle, DOC-ECI”). 2. POSDRU, financing contract POSDRU/615/S/19.

References: