Improving Orienteering-based Tourist Trip Planning with Social Sensing

Fabio Persia

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy

Giovanni Pilato

Institute for High Performance Computing and Networking, National Research Council of Italy, Palermo

Mouzhi Ge

Faculty of Informatics, Masaryk University, Czech Republic

Paolo Bolzoni

School of Computing, Dublin City University, Ireland

Daniela D'Auria

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy

Sven Helmer

Department of Informatics, University of Zurich, Switzerland

Abstract

We enhance a tourist trip planning framework based on orienteering with category constraints by adding social sensing. This allows us to customize a user's experience without putting the burden of preference elicitation on the user. We identify the interests of a user by analyzing their Tweets and then match these interests to descriptions of points of interests. For this analysis we adapt different schemes for social sensing to the needs of our orienteering context and compare them to find the most suitable approach. We show that our tech-

Email addresses: fabio.persia@unibz.it (Fabio Persia), giovanni.pilato@icar.cnr.it (Giovanni Pilato), mouzhi.ge@muni.cz (Mouzhi Ge), paolo.bolzoni@dcu.ie (Paolo Bolzoni), daniela.dauria@unibz.it (Daniela D'Auria), helmer@ifi.uzh.ch (Sven Helmer)

Author Accepted Manuscript (AAM) of the paper:

Persia, F., Pilato, G., Ge, M., Bolzoni, P., D'Auria, D., & Helmer, S. (2020). Improving orienteering-based tourist trip planning with social sensing. Future Generation Computer Systems, 110, 931-945. https://doi.org/10.1016/j.future.2019.10.028

nique is fast enough for use in real-time dynamic settings and also has a higher accuracy compared to previous approaches. Additionally, we integrate a more efficient algorithm for solving the orienteering problem, boosting the overall performance and utility of our framework further, as demonstrated by the positive user satisfaction received by real users.

Keywords: `social sensing`, `orienteering`, `semantic mapping`, `semantic similarity`

---

1. Introduction

In tourist trip planning, it is generally not useful to determine shortest paths from a source location to a destination location. Instead, it is more important to discover routes covering the most attractive points of interests (POIs), ideally matching the personal tastes of a tourist [1]. A common and well-known way to model tourist trip planning is by mapping it onto an *orienteering problem*. The orienteering problem (OP) considers the route network as an edge-weighted graph with a weight and a score for each node. The goal is to discover a path from a starting node to a destination node that maximizes the total score while staying within a certain time budget. Having its roots in the longest path problem, OP is an NP-hard problem, which means that computing exact solutions is only possible for small problem instances.

There are more efficient algorithms for OP based on heuristics or approximations, which provide good answers in a fraction of the time it needs to compute the optimal answer. In the context of tourist trip planning this is fine, as tourists are already content with a route close to the optimal one. However, these approaches assume that the user preferences are given as input parameters to an algorithm, but do not specify where these preferences come from. In the case of category-based OP, a user would have to specify which categories of POIs they like and to which degree they like them, otherwise an algorithm cannot personalize the route.

The proliferation of numerous online social networks, such as *Facebook*,

LinkedIn, and *Twitter*, has now made it possible to incorporate social elements into a variety of data-centric applications [2]. The sheer amount of information found in social networks is huge and on top of that it is heterogeneous, often containing complex linkage structures. *Social sensing* is an approach to analyze user-generated content in social networks in such a way as to make it usable for a range of applications [3]. We propose a method based on social sensing to obtain personalized POI scores without explicitly asking users about their specific tastes.

More specifically, we look at the Twitter posts of users and compute semantic distances between a user profile and the description of POIs. In this way we obtain a list of POIs together with personalized scores. After normalizing these scores, they can be fed directly into an OP solver together with the route graph to generate a custom-tailored trip without having to modify the original OP algorithm. In summary, we focus on the following aspects and make a number of contributions:

- We propose an accurate social sensing technique for orienteering by combining schemes such as *Latent Dirichlet Allocation (LDA)*, *Latent Semantic Analysis (LSA)*, the *spaCy*[1] tool, and the *RAKE* algorithm [4];
- For solving the associated OP, we integrate an efficient anytime algorithm that was developed recently [5], making it possible to show incrementally improving solutions to users;
- We evaluate our approach empirically using real-world data sets of Tweets and descriptions of POIs in Italian; we opted for Italian rather than English due to two reasons; first, Bolzano being an Italian city, most of the descriptions of its POIs are in Italian; second, as there are significantly fewer applications in Italian, a framework in this language offers more novelty;

[1] https://spacy.io/

- We demonstrate the effectiveness of our approach in terms of user satisfaction via a user study in which users get to compare and choose from solutions generated with and without social sensing;

The remainder of the paper is organized as follows: in Section 2 we provide a motivating example to illustrate the intuition of our work, whereas in Section 3 we review the technologies that are used in this paper. In Section 4 the overall process is presented, and the experimental evaluation is exhibited in Section 5. Eventually, Section 6 concludes the paper and outlines the future work.

2. Motivating Example

We now introduce a motivating example to illustrate the usefulness of social sensing for orienteering. We take POIs from the city of Bolzano: Figure 1 shows a graph with a source node s (*Waltherplatz*), a destination d (*Via Rovigo*), and three POIs, p_1 (*Cathedral of Bolzano*), p_2 (*Christ of the King Church*), and p_3 (*Dr. Friedrich Tessmann Library*), along with their distances expressed in minutes. We simplify the graph to keep it readable: all visiting times are supposed to be 60 minutes, all POIs belong to the same category, and we also omit some edges.

Let us assume that Elon Musk (co-founder, CEO, and product architect of Tesla, Inc., as well as one of the top-200 most followed twitter accounts) would like to spend 3 hours (i.e., 180 minutes) in Bolzano, starting from Waltherplatz (s) and ultimately arriving at a friend's apartment (located at d). The algorithm in [6], without social sensing, would suggest the itinerary $I_1 = \langle s, p_1, p_2, d \rangle$, assuming all the scores to be the same. In fact, if all the other parameters are the same, the algorithm suggests the itinerary with the best centrality, that is the one containing the nodes with the shortest distance to s and d (for more details on the algorithm, see Section 4.6).

On the other hand, the new algorithm including *social sensing* extracts from Elon Musk's tweets interesting topics such as "project", "Tesla", and "physics", that implies that p_3 (the *Dr. Friedrich Tessmann Library*, one of the most

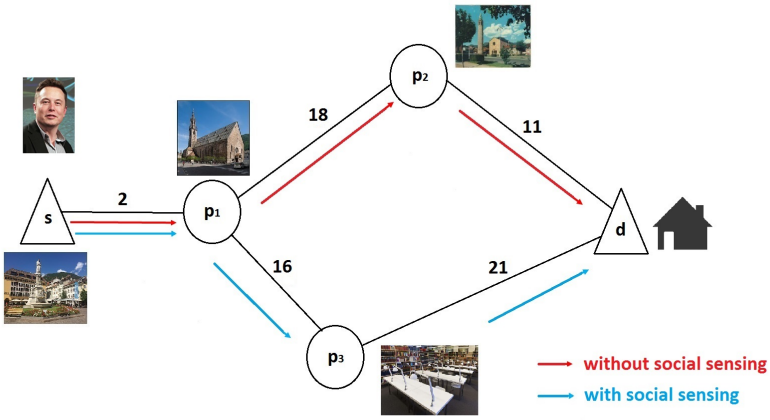


Figure 1: Route Orienteering for Elon Musk

modern and well-stocked libraries in Italy), is implicitly ranked higher than p_2 by Elon Musk. As a result, its score will be higher than the one of p_2, and consequently the algorithm suggests the itinerary $I_2 = \langle s, p_1, p_3, d \rangle$; even if it is longer than I_1, it is likely to be appreciated more by this particular twitter user. So, the most interesting path for him in Bolzano would start at Waltherplatz, go to the Cathedral in two minutes, spend an hour there, then arrive at the Dr. Friedrich Tessmann Library sixteen minutes later, stay an hour there, and reach the friend's apartment in Via Rovigo 21 minutes later. This path has a total length of 159 minutes, leaving more than 20 minutes to the time constraint of three hours.

3. Related Work

The *orienteering problem* was initially defined by Tsiligrides in [7]. Since it is an NP-hard problem, algorithms computing the exact solutions via branch and bound or dynamic programming techniques [8, 9] can only solve small instances

of the problem. As a consequence, most of the used approximation algorithms and heuristics adopt a two-step approach of partial path construction [10, 7] and (partial) path improvement [11, 12, 13]. Additionally, other approaches in the literature exploit meta-heuristics such as genetic algorithms [14] and neural networks [15]. There are also a few approaches that consider the orienteering problem generalized by categories. These approaches are based on clustering the nodes [16], applying an optimized best-first search [5], and using probabilistic methods [6]. However, none of these techniques consider using social sensing for eliciting user preferences.

In order to integrate social network information into the solution of an orienteering problem, the first step is to process the *Online Social Network (OSNs)* data. There are two general categories of methods that can extract, collect, and mine the information from *OSN* data: *machine learning* approaches and *non-machine learning* approaches. An example of a *machine learning* approach is [17], which uses a range of features including language, hashtags, and geographical locations of their social ties to infer a user's nationality on Twitter; thus, it can be classified as a method based on *user profile*. Furthermore, Bentwood [18] uses a topology of influence to quantify the impact of social media, while Tinati et al. [19] uses a social network graph based on retweet behavior to identify key players in a conversation. They describe a new approach to classify users within the Twitter service and use a topology of influence. Moreover, D'Avanzo et al. [20] introduce a pipeline, implemented as a web service, to allow decision makers to monitor Twitter's sentiment regarding trends based on Google Trends. It also enables users to choose geographic areas for their monitors. In addition to the positive/negative sentiments on Google Trends, the pipeline offers the ability to view, on the same dashboard, the emotions that Google Trends triggers in the Twitter population. For the *non-machine learning* approaches, Gianvecchio et al. [21] propose an *entropy-based* classification method to identify the chat bots in Yahoo Chat, while Sarwar et al. [22] exploit *user-based* collaborative filtering to predict a test user's interest in a test item based on rating information from similar user profiles.

We now look at some well-known techniques and tools used by us for sensing the interests of user from their social network profiles. More specifically, in subsection 3.1 we show the approach built on the LSA-based Semantic Space, in subsection 3.2 the one built on the LDA-based Topics Space, in subsection 3.3 the one based on an effective NLP tool named spaCy, and in subsection 3.4 the one based on the RAKE algorithm.

3.1. LSA-based Semantic Space

The Latent Semantic Analysis (LSA) technique is a well-known method useful for obtaining a vector encoding of words and their semantics [23]. LSA has been widely used for simulating different human cognitive phenomena [24][25]. Given a set of M terms and N documents, the LSA technique is based on a term-document co-occurrence $M \times N$ matrix $\mathbf{A}$, whose generic element $[a_{i,j}]$ is a function of the number of times a term occurs in a document. In particular, we have considered the classical "Term frequency - Inverse document frequency" *Tf-Idf* approach for encoding a generic element $[a_{i,j}]$ of $\mathbf{A}$. Given a word w_i and a document d_j, let $f_{i,j}$ be the number of occurrences of w_i in d_j and let n_{w_i} be the number of times the word w_i appears at least once in a document of the collection. Then, a generic element $[a_{i,j}]$ of the matrix $\mathbf{A}$ is:

$$[a_{i,j}] = (1 + log(f_{i,j})) \cdot log\left(\frac{N}{n_{w_i}}\right) \tag{1}$$

The matrix $\mathbf{A}$ can be decomposed into the product of three matrices, $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$ by exploiting a factorization procedure known in literature as "Singular Value Decomposition" (SVD)[23].

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \tag{2}$$

$\mathbf{U}$ is a column-orthonormal $M \times N$ matrix, $\mathbf{V}$ is a column-orthonormal $N \times N$ matrix and $\mathbf{\Sigma}$ is a $N \times N$ diagonal matrix. The elements of $\mathbf{\Sigma}$ are named "singular values" of $\mathbf{A}$. The columns of $\mathbf{U}$ are named left singular vectors of $\mathbf{A}$ and the columns of $\mathbf{V}$ are named right singular vectors of $\mathbf{A}$. It may be assumed, without loss of generality, that the singular values of $\mathbf{A}$ are ranked in

decreasing order, since Singular Value Decomposition also holds if we perform permutations of singular values along with the corresponding rows in $\mathbf{U}$ and $\mathbf{V}$ [25].

Let R be an integer with $0 < R < N$, and let $\mathbf{U}_R$ be the $M \times R$ matrix extracted from $\mathbf{U}$ by neglecting the last $N - R$ columns, let $\mathbf{\Sigma}_R$ be the matrix extracted from $\mathbf{\Sigma}$ by neglecting the last $N - R$ rows and the last $N - R$ columns; let $\mathbf{V}_R$ be the $N \times R$ matrix extracted from $\mathbf{V}$ by neglecting the last $N - R$ columns. Then, it can be shown that:

$$\mathbf{A}_R = \mathbf{U}_R \Sigma_R \mathbf{V}_R^T \tag{3}$$

$\mathbf{A}_R$ is an $M \times N$ matrix of rank R, and it is the best rank R approximation of $\mathbf{A}$, among the $M \times N$ matrices, with respect to the Euclidean metric. The i-th row of the matrix $\mathbf{U}_R$ is the numerical encoding of the i-th word. The columns of the $\mathbf{U}_R$ matrix represent the R independent dimensions of the induced semantic space. Moreover, it is possible to extract the words which best describe these independent dimensions by computing a score $Sc(w_i)$ for each word w_i by applying the formula [25, 26]:

$$Sc(w_i) = \sum_{j=1}^{R} \sigma_{ij} |u_{ij}| \tag{4}$$

with $\sigma_{ij} \in \Sigma_R$ and $u_{ij} \in U_R$.

3.2. LDA-based Topics Space

Latent Dirichlet Allocation (LDA) is an unsupervised method for computing a Bayesian probabilistic model of text corpora, with the aim of finding topics in documents [27].[2] In a nutshell, it assumes that the documents in a corpora were created using a generative process (for a gentle introduction to LDA and generative probabilistic topic models, see [29]). Put simply, we assume that every document follows a certain topic distribution (e.g. 70% topic A and

[2]LDA extends the probabilistic Latent Semantic Analysis (pLSA) approach [28], which, even though its name looks similar to LSA, follows a completely different procedure.

30% topic B). Furthermore, a topic is specified as a distribution over a fixed vocabulary (i.e., the words contained in a document collection). Generating a document is split into two stages: first we (randomly) choose a distribution over topics for the document and then every word is randomly determined by the distribution over the vocabulary while following the topic distribution of the document. The main issue with this process is that we know neither the topic distributions of the documents nor the vocabulary distributions of the topics, these are hidden variables. LDA turns this around by searching for the most probable hidden parameters that could have generated an observed document collection. More formally, this is done in the following way. The number of topics, K, is fixed beforehand and every topic $\beta_k \in \beta_{1:K}$ is a distribution over the vocabulary. Now we turn to the document collection, which contains a total of D documents. The percentage of topic k in document d is expressed by $\theta_{d,k}$, θ_d defines the percentages of all the topics in this document. The distribution $z_{d,n}$ determines which topic is assigned to the word n in document d, z_d stands for all these assignments in d. Last, but not least, $w_{d,n}$ represents the n-th word in document d, w_d denotes all the observed words in d. Putting all this together, we can now describe the whole process of generating documents using a joint probability distribution:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p(z_{d,n})|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n}) \right) \tag{5}$$

On a technical note: the different distributions above are modeled using symmetric Dirichlet distributions. The parameters β, θ, and z are called latent variables, as they are not directly observable. Given the observations (in the form of word frequencies in the documents), we reverse-engineer the generative process by computing the posterior distributions of the latent variables, which boils down to solving

$$\mathbf{p}(\beta_{\mathbf{1:K}}, \theta_{\mathbf{1:D}}, \mathbf{z_{1:D}}|\mathbf{w_{1:D}}) = \frac{\mathbf{p}(\beta_{\mathbf{1:K}}, \theta_{\mathbf{1:D}}, \mathbf{z_{1:D}}, \mathbf{w_{1:D}})}{\mathbf{p}(\mathbf{w_{1:D}})} \tag{6}$$

The process of solving this equation is intractable from a computational point of view. However, its solution can be approximated through variational inference and Gibbs sampling [30, 27, 31].

3.3. SpaCy Tool

For this method, we used two natural language processing (NLP) tools, namely fastText [32] and spaCy [33], expanding and improving the spaCy baseline approach we applied in [34]. Our technique is composed of two steps: preprocessing documents with the help of fastText and comparing them via spaCy.

Before starting with the preprocessing, we remove all links to webpages, references to other users, and hashtags from our collection of Tweets. We then feed the words contained in the Tweets and the POI descriptions into fastText, which computes distributed word representations, or word vectors, for them [35, 36]. This is a popular technique in machine learning for uncovering subsymbolic meanings, such as word analogies. We utilized a pre-trained word vector encoding for Italian provided by fastText [32], which was trained on Common Crawl and Wikipedia. As reported by fastText, the parameters used for this training are as follows: 300-dimensional vectors, a continuous bag of words (CBOW) model with position weights, character n-grams of length five, a window size of five, and ten negative examples [32, 36].

We use a simple strategy for determining the similarity between Twitter profiles and POI descriptions. Assuming that each POI is described by (at least) one sentence, we compute the similarity between the word vectors describing the Tweets and the word vector of the sentence describing a POI via the cosine measure. If the similarity between a Tweet and a POI is greater than a threshold θ, the Tweet matches the POI (the value of θ is set to 0.85, which was determined experimentally). For measuring the similarity of the whole Twitter account to a POI, we take the number of matching Tweets and divide it by the total number of Tweets for this account. If a POI is described by more than one sentence, we only consider the sentence with the highest similarity. We repeat this procedure

for every POI, obtaining an ordered list of POIs according to the semantic similarity with the user posts.

In our previous work we showed that a social sensing approach based on LDA and LSA is able to achieve an accuracy similar to a simpler baseline version of the spaCy approach, but at a much faster rate [34]. Here, we build on top of this by optimizing the spaCy approach (see scheme 5 in Section 4.3.3), decreasing the number of operations per user it needs.

3.4. Rapid Automatic Keyword Extraction (RAKE)

The RAKE algorithm [4] has been introduced by Rose et al. with the intention to get a general and efficient algorithm for extracting keywords from documents. Keywords are defined as sequences of one or more words capable of supplying a concise representation of the subject of a document [4].

RAKE is based on the observation that keywords often contains multiple words, but they seldom contain stop words in the middle. In addition to the documents, the method takes as input a set of of stop words, e.g. for English these are words such as *"the"*, *"of"*, or *"and"*, and a set of phrase and words delimiters like blanks or punctuation marks. The stop words and delimiters are used to partition the text into word sequences, with each sequence being considered a candidate keyword.

The algorithm is unsupervised, domain-, and language-independent. In fact, the original paper [4] also describes a method to automatically generate a list of stop words, starting with a set of documents.

Once the list of the candidate keywords has been identified, each word w_i present in the text is scored by using the function:

$$s_{w_i} = \frac{degree(w_i)}{occ(w_i)} \tag{7}$$

where $occ(w_i)$ is the number of occurrences of w_i in the text, and $degree(w_i)$ is the number of times that the word w_i co-occurs with other words in a candidate keyword sequence. A higher value of word $degree(w_i)$ means that w_i occurs in

long candidate keywords. The index i ranges from one to the number of unique words occurring in the text.

Once the score of each word in the text has been computed, a grade is associated with each candidate keyword by summing the scores of the words that constitute it. The candidate keywords are then sorted by descending value of their grades and the top third are selected as keywords of the text [4].

The algorithm can also be enabled to detect keywords with stop words in the middle. In this case, once the document in question is parsed and a set of words is found, the algorithm looks for pairs of possible keywords with a stop word in the middle that appear at least twice in the same document and includes them among the list of keyword candidates. This extra step allows us to obtain keywords like *"set of natural numbers"*, where the presence of the stop word *"of"* would have prevented us from finding it.

Experiments reported in [4] show that RAKE is an efficient algorithm, while at the same time achieving a good ratio of precision and recall. In this paper, RAKE has been used as one of the schemes to extract keywords from user tweets.

4. Orienteering based on Social Sensing

In this section, we show the overall process we followed for building an orienteering framework supported by social-sensing (Figure 2). It consists of six layers: *OSN crawler*, *tweets extraction and preprocessing*, *semantic space mapping*, *semantic similarity matrix building*, *ranking computation*, and *integration with the orienteering approach*.

The output of the OSN crawler consists of a set Λ of collected data related to user sessions on a particular OSN. As a result, in this context we collect all the *user tweets*, which are further processed in the following steps (Figure 2) of the pipeline, in order to detect the *user interests* in *POIs*.

Then, the *tweets extraction and preprocessing step* aims at cleaning the text of the tweets before further processing. Afterwards, the *semantic space mapping*

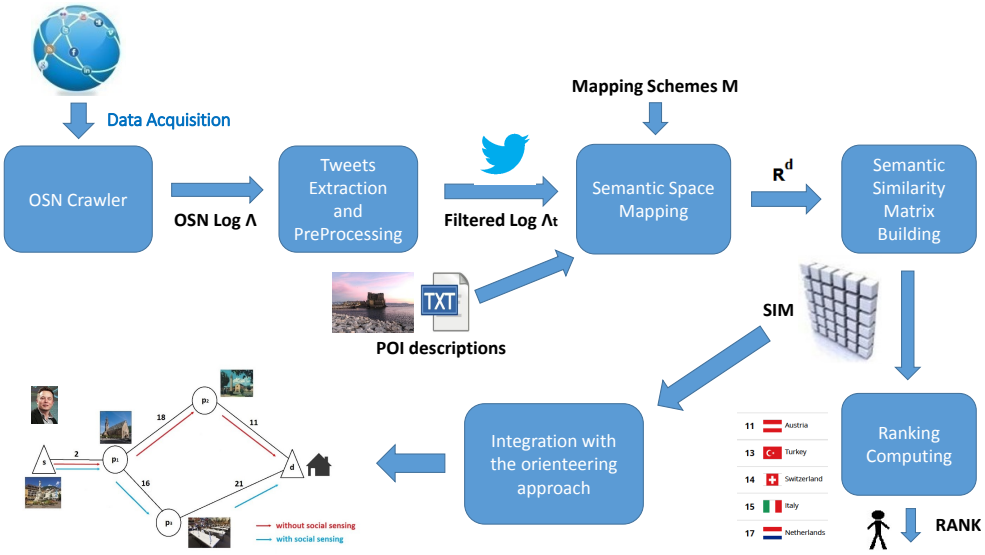


Figure 2: The Overall Process

phase is applied to map both the filtered tweets and the POI descriptions into a joint semantic space $\mathbf{R^d}$; the details of this method are explained later on.

With the joint semantic space, the overall *semantic similarity matrix SIM* between *users* and *POIs* is computed; a generic element $s_{ij} \in SIM$ stands for the *user interest* of user i in POI j, computed by analyzing the similarity of the *user tweets* and the *POI descriptions*. Afterwards, two further processing steps are possible; on the one hand, a specific *RANK* matrix holding the user interests in the POIs ordered by descending values is extracted; this is done to ease the process of extracting the *Top-k* list of preferences for each user, where $0 < k < n$, n being the total number of considered POIs. The *RANK* matrix can be used in an *offline* setting as an input for applications such as POI visualization or recommender systems.

On the other hand, the *SIM* matrix can be exploited for customizing the scores of the nodes as used in an algorithm for solving the orienteering problem.

We provide customized paths that also take into account the *user interest* in the POIs, implicitly extracted from a user's tweets. As a result, the score of each POI (Figure 2) can be considered as a function of the similarities among the topics of the user tweets and the topics of the POI descriptions. The detailed procedure is described in Subsection 4.6.

4.1. OSN Crawler

We start off by considering a finite set of atomic user actions in an OSN along the lines of [37] (Table 1). We assume that the set of observable actions in the domain is finite, since we deal with sequences of time-stamped events. Each action can be associated with a different action symbol in S, the user to whom it belongs, a related timestamp, and other useful attributes for characterizing actions. In this paper we analyze *Twitter posts* and focus on the "e" symbol of Table 1. Thus, we store the *content of tweets* as additional information for characterizing the action. We base the definition of our *OSN Log* on the one provided in [37, 38, 39].

OSN Log: A *log entry* is a tuple $\lambda = (s, user, ts, att_1, ..., att_k)$, where $\lambda.s \in \mathcal{S}$ is the action symbol associated with the event, $\lambda.user$ is the user with which the action is associated, $\lambda.ts$ is the related timestamp and $\lambda.att_1 ... \lambda.att_k$ are other attributes optionally associated with the observation (e.g. the content of a tweet, or an IP address). An *OSN Log* Λ is then a finite sequence of log entries λ_i ordered by timestamp.

In this context, the task of the *OSN Crawler* is the collection of real datasets, such as the one described in subsection 5.2.1, using the Twitter APIs with the default access level.

4.2. Tweets extraction and preprocessing

Given a user Twitter ID, we extract their tweets. For our approach, we only use the tweet text content, which is preprocessed before being utilized to build a data-driven conceptual space. Stop-words are filtered out, and links are removed before processing the text, since they often hide off-topic posts or

Table 1: Action/Symbol

Symbol	**Action**
a	login
b	message received
c	message sent
d	friend approved
e	status wall post
f	link shared story
g	video shared story
h	status update
i	added picture
j	published link
k	pictured shared story
l	youtube video shared
m	youtube created story
n	link app created story
o	tagged in a picture
p	mobile status update
q	checkin
r	likes a page
s	logout

even spam. Abnormal sequences of characters are discarded and the twitter text is then lemmatized. In this context the *OSN Crawler* can be used either for retrieving tweets satisfying a query composed of keywords, or to download the most recent tweets of a given twitter user ID. The first step of the followed procedure is the text extraction. More specifically, for each Twitter account x we retrieve all the most recent user tweets via the Twitter APIs and then store them. As a result, in our case study the maximum number of tweets detected per user is about 3200. We also detect the user's followers and friends, and save them for possible future use. Afterwards, we identify the language of the tweets; if less than 60% of the user tweets are in Italian or the user tweets in Italian are less than 20, we ignore the Twitter account. Otherwise, we extract from the tweets only the textual content (getting rid of all the other information).

4.3. Semantic scoring of POIs according to the user interests

Given the user Twitter ID, we have set up a set of approaches that can help in finding a semantic similarity between the user tweets and the natural language description of each POI. In particular, we defined six different schemes exploiting a semantic space induced from user tweets by Latent Semantic Analysis (LSA) [23][40], a topic space induced by the user tweets from Latent Dirichlet Allocation (LDA) [27], the *spaCy* tool, and the RAKE algorithm [4]. As discussed in Section 3, LSA, LDA, spaCy, and RAKE algorithm are techniques that can also be used for profiling users and to extract the main topics in tweets [26][41]. We explain all the different approaches for getting the semantic scoring of POIs according to the user interests in Sections 4.3.1 to 4.3.4.

4.3.1. Semantic scoring schemes 1 and 2

The weights identified by the schemes with numbers 1 and 2 are respectively:

1. the *maximum* value of Jaccard similarity between topics induced by using LSA and LDA from the tweets of the user, and the topic descriptions;
2. the *average* value of Jaccard similarity between topics induced by using LSA and LDA from the tweets of the user, and the topic descriptions.

In order to obtain these weights, we apply the following procedure. First, we remove from the textual content of the tweets any links, references to other accounts, and hashtags (since the POI descriptions do not contain any of such elements); then, we also remove stopwords. Consequently, on such filtered tweets of the *i-th* user (that we identify as set Λ_{ti}) we execute LSA, thus obtaining the first 20 *conceptual axes* along with their descriptive words (10 per axis). Then, we also apply LDA on Λ_{ti}, thus obtaining 20 topics along with their descriptive words (10 per topic). As a result, by applying *LSA* and *LDA* on Λ_{ti}, we obtain a file containing 40 rows; each of them has 10 words, describing the possible basic user interests (U_I). On the other hand, we remove the stopwords from the POI descriptions; for each descriptive sentence of a POI, we compute the Jaccard similarity (Intersection/Union) with each of the 40 rows of the user as well as the maximum value obtained for each descriptive sentence, keeping also track of all the Jaccard values. More formally, we define:

- n_s as the number of sentences of the POI description;
- n_{U_i} as the total number of *conceptual* descriptions of user interests (40 in this case).

Consequently, the computation of the Jaccard similarities implies the construction of the J matrix; more specifically, in Equation 8 the *Jaccard* function is defined on the domain $\{1, ..., n_s\} \times \{1, ..., n_{U_i}\}$, and has the J matrix as co-domain.

$$Jaccard : \{1, ..., n_s\} \times \{1, ..., n_{U_i}\} \rightarrow J \tag{8}$$

Eventually, for each tuple $(user, POI)$, we compute:

- the average of the maximum values (*weight 1*, Equation 9);
- the average of all the Jaccard values (*weight 1*, Equation 10).

$$w_{ij}^{(1)} = avg\{\max_k\{J_{k*}\}\}, \qquad \forall i = 1, ..., m; \forall j = 1, ..., n; \forall k = 1, ..., n_s; \tag{9}$$

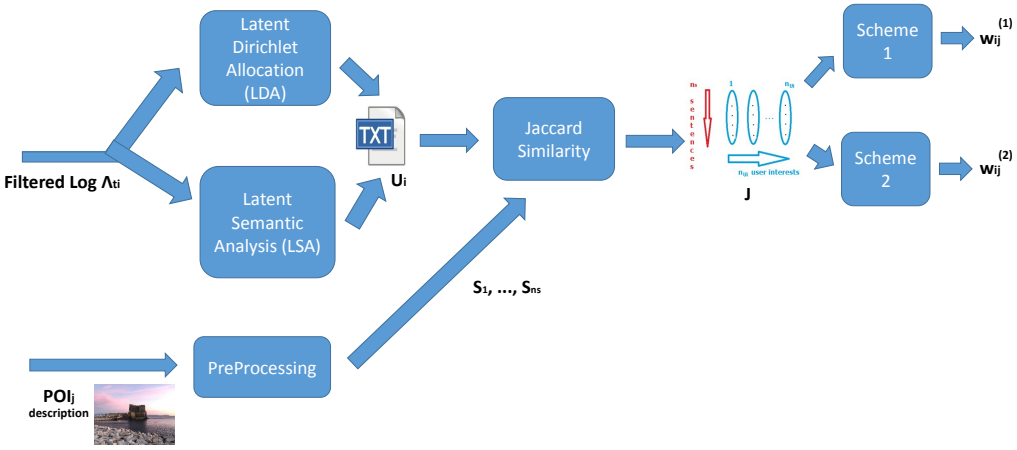


Figure 3: Scheme 1 and 2, for each tuple ($user_i$,POI_j)

$$w_{ij}^{(2)} = avg\{J_{kl}\}, \qquad \forall i = 1, ..., m; \forall j = 1, ..., n; \forall k = 1, ..., n_s; \forall l = 1, ..., n_{U_i} \tag{10}$$

4.3.2. Semantic scoring schemes 3 and 4

The weights identified by the schemes with numbers 3 and 4 are, respectively:

1. the *maximum* value of Jaccard similarity between the user tweets and the topic descriptions;
2. the *average* value of Jaccard similarity between the user tweets and the topic descriptions.

The procedure to obtain these weights is described in the following. First, we remove from the textual content of the tweets links, references to other accounts, hashtags (since in the POI descriptions there are no such elements), and the stopwords. Then, we also remove the stopwords from the POI descriptions.

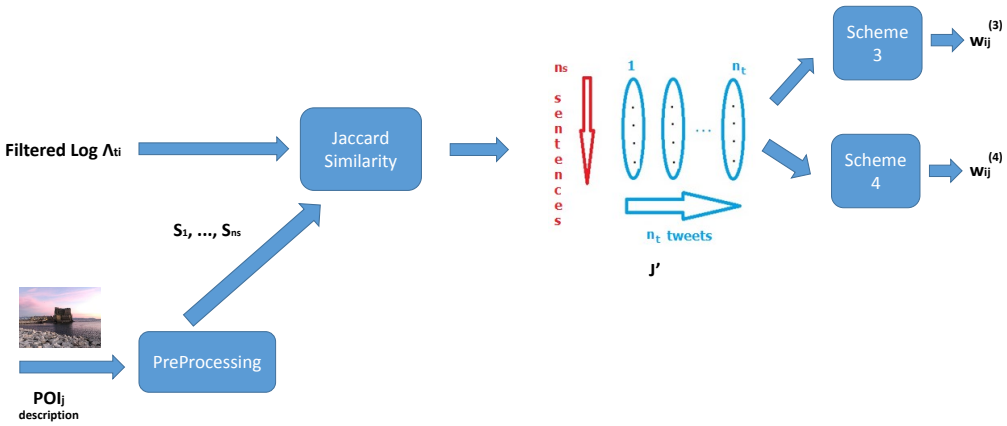


Figure 4: Scheme 3 and 4, for each tuple ($user_i$,POI_j)

Consequently, for each descriptive sentence of the POI we compute the Jaccard similarity (Intersection/Union) with every user tweet. Formally, assuming n_t to be the number of user tweets, we build the following matrix J'.

$$Jaccard : \{1, ..., n_s\} \times \{1, ..., n_t\} \rightarrow J' \tag{11}$$

Eventually, we compute $w^{(3)}$ (Equation 12) and $w^{(4)}$ (Equation 13) for each tuple $(user, POI)$ as follows:

$$w_{ij}^{(3)} = avg\{\max_k\{J'_{k*}\}\}, \forall i = 1, ..., m; \forall j = 1, ..., n; \forall k = 1, ..., n_s \tag{12}$$

$$w_{ij}^{(4)} = avg\{J'_{kl}\}, \qquad \forall i = 1, ..., m; \forall j = 1, ..., n; \forall k = 1, ..., n_s; \forall l = 1, ..., n_t \tag{13}$$

4.3.3. Semantic scoring scheme 5

The weight identified by scheme 5 is explained in the following:

1. all the tweets of a user taken together are treated as a single document, whereas every description of a POI forms its own document. All documents are encoded with fastText and we compute their similarity with the help of the cosine measure provided by spaCy.

More specifically, we follow the listed procedure:

1. we encode the concatenated tweets of the user i as a vector with fastText, exploiting the "it_vector_wiki_lg" model, obtained from the Italian version of Wikipedia. As a result, we get a single vector (enc_tweets) in a 300-dimensional space, which is the encoding of all the tweets of user i;
2. we follow the same procedure as the one explained in step 1 for encoding each POI description. Consequently, we obtain the matrix enc_POIs, whose generic column j represents the encoding in a 300-dimensional space of the j-th POI;
3. we compute for each Twitter user the semantic distance (the *cosine*) between all the user tweets (as a single document) and the POI descriptions. More formally, remembering that m is the total number of users, and n the total number of POIs, we define the weight 5 ($w^{(5)}$) as follows:

$$w_{ij}^{(5)} = cos(enc\_tweets_{i*}, enc\_POIs_{*j}) \qquad \forall i = 1, ..., m; \forall j = 1, ..., n \tag{14}$$

We can sort these weights by similarity in descending order, obtaining the final ranking.

4.3.4. Semantic scoring scheme 6

The weight identified by scheme 6 is defined by

- the similarity between descriptive keywords extracted by the RAKE algorithm and the description of the POI.

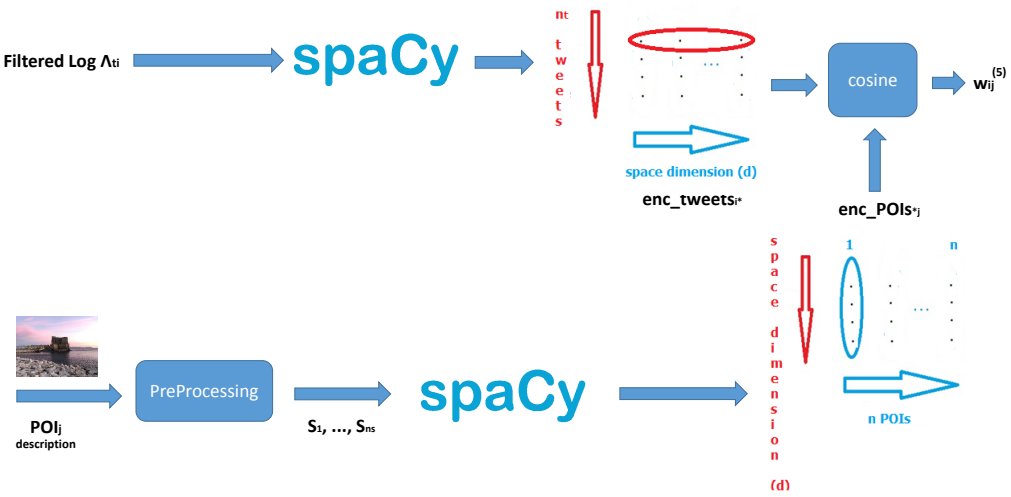


Figure 5: Scheme 5, for each tuple ($user_i$,POI_j)

We proceed as follows.

1. We compute the enc_POIs matrix as described in subsubsection 4.3.3;
2. we use the RAKE algorithm [4] to extract the n_{key} keyphrases (each of them identified by key_k in Equation 15) from the whole set of tweets of the i-th user;
3. each of the keyphrases is associated with a $score$ ($score_k$);
4. eventually, we build weight 6 for each tuple $(user_i, POI_j)$ as follows:

$$w_{ij}^{(6)} = \sum_{k=1}^{n_{key}} score_k \cdot cos(key_k, enc\_POIs_{*j}) \qquad \forall i = 1, ..., m; \forall j = 1, ..., n \tag{15}$$

4.3.5. Final remarks about semantic scoring schemes

As shown in the previous sections, all the described semantic scoring schemes map the filtered user tweets as well as the POI descriptions into $\mathbf{R^2}$, i.e., $d = 2$

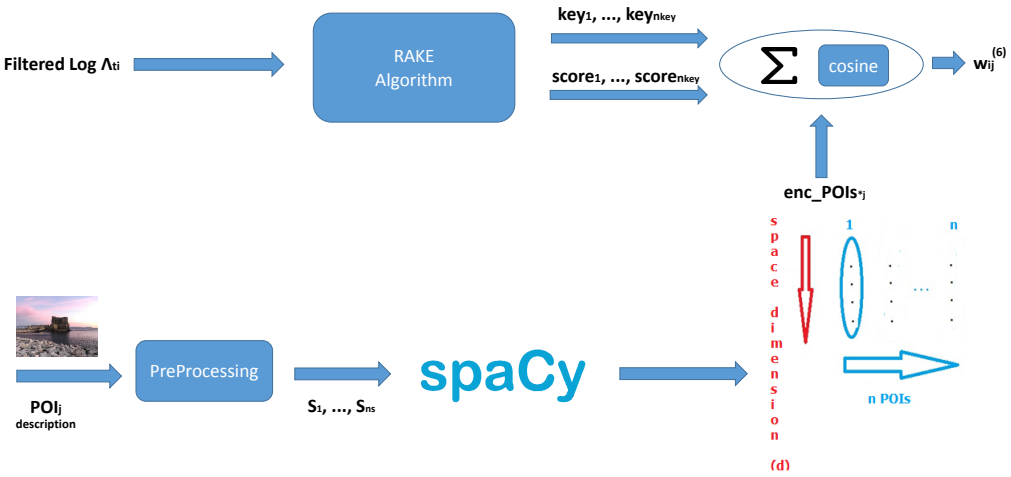


Figure 6: Scheme 6, for each tuple ($user_i$,POI_j)

(Figure 2). However, possible future semantic scoring schemes mapping into $\mathbf{R^d}$, $d > 2$, could also be integrated into the framework.

4.4. Semantic Similarity Matrix Building

In this step we combine the weights of a scoring scheme, which we discussed earlier, into a similarity SIM; more specifically, a generic item of the SIM matrix stands for the interest of user i in POI j, whereby the higher the value, the greater the interest. As a result, we define a similarity matrix for each of the defined schemes as follows:

$$SIM_{ij}^{(k)} = w_{ij}^{(k)} \qquad \forall i = 1, ..., m; \forall j = 1, ..., n; k = 1, ..., 6 \tag{16}$$

In Subsection 4.5 we show how to use the ranking computing functionality in an offline setting, whereas in Subsection 4.6 we demonstrate how to use the similarity matrix to compute customized node scores.

4.5. Ranking Computing

Now we show how to build a $RANK$ matrix starting from a SIM matrix. The $RANK$ matrix is useful as input for applications such as POI visualization, recommender systems, or data analytics; in fact, in these contexts it is convenient to use a matrix whose rows are sorted in a descending order, thus explicitly listing the most popular POIs in the first positions. More formally, we first define the *sort* function (Equation 17), which orders a vector v in descending order.

$$sort : \{v_1, ..., v_m\} \rightarrow \{v'_1, ..., v'_m\}, | \quad v'_i \geq v'_{i+1}, \forall i = 1, ..., m-1; \tag{17}$$

Consequently, Equation 18 defines the $RANK$ matrix as

$$RANK_{i*} = sort\{SIM_{i*}\} \qquad \forall i = 1, ..., m; \tag{18}$$

whose generic items stand for the interest of the user i in the POI j, sorted in descending order. Clearly, for each generic item $RANK_{ij}$ we also keep track of the ID of the related POI_j. However, since the complexity of building the $RANK$ matrix is $O(m \cdot n \cdot log(n))$, it is better to exploit this functionality in an offline setting (where the matrix has not to be computed too frequently), rather than in a real-time setting.

4.6. Integration with the Orienteering Approach

Before describing the algorithm for solving the orienteering problem (OP) used in [5], and the way we combine it with social sensing, we define the formal model that we utilized for tourist trip planning. Since there is no obligation to visit all the nodes in the OP, it is also called *Selective Traveling Salesman Problem (STSP).*

We assume a set P of points of interest (POIs) p_i, $1 \leq i \leq n$. The POIs, together with a starting and a destination node, denoted by s and d, respectively, are connected by a complete, metric, weighted, undirected graph $G = (P \cup$

$\{s, d\}, E)$, whose edges, $e_l \in E = \{(x, y)|x, y \in P \cup \{s, d\}\}$, connect them. Each edge e_l has a cost $c(p_i, p_j)$ that signifies the duration of the trip from p_i to p_j, while every node $p_i \in P$ has a cost $c(p_i)$ that denotes its visiting time. Each POI belongs to a certain category, such as *museums*, *restaurants*, or *galleries*. The set of n_{cat} categories is denoted by K and each POI p_i belongs to exactly one category $\kappa_k \in K, 1 \leq k \leq n_{cat}$. Given a POI p_i, $cat(p_i)$ denotes the category p_i belongs to and $score(p_i)$ denotes its score or reward, with higher values indicating higher interest to the user. Finally, users have a certain maximum time in their budget to complete the itinerary, denoted by t_{max}.

Furthermore, let P^* be the ordered set of vertices in the optimal path, and define a binary variable x_{ij} taking the value 1 if and only if p_i and p_j are two consecutive vertices of P^*. Additionally, let y_{ik} be another binary variable taking the value 1 if and only if POI p_i belongs to category κ_k, and $max_{\kappa_k}, \forall \kappa_k \in K$ represents the maximum number of POIs belonging to category κ_k allowed in P^*.

Consequently, the orienteering problem for tourist trip planning is formally modeled as follows, with $1 \leq i, j \leq n$ and $1 \leq k \leq n_{cat}$:

$$\textbf{max} \sum_i score(p_i) \sum_j x_{ij}$$

subject to

$$\sum_i x_{si} = \sum_i x_{id} = 1 \tag{19}$$

$$\sum_i x_{il} = \sum_i x_{li} \leq 1, \quad \forall l : 1 \leq l \leq n \tag{20}$$

$$\sum_i \sum_j (c(p_i) + c(p_i, p_j)) \cdot x_{ij} \leq t_{max} \tag{21}$$

$$\sum_k y_{ik} = 1, \quad \forall i : 1 \leq i \leq n \tag{22}$$

$$\sum_i \sum_j x_{ij} \cdot y_{ik} \leq max_{\kappa_k}, \quad \forall k : 1 \leq k \leq n_{cat} \tag{23}$$

$$x_{ij} \in (0,1); \quad i \neq j$$

More specifically, the constraint in Equation 19 specifies that both s and d belong to P^* as the first and last POI, respectively; additionally, constraint (20) defines the flow conservation conditions for the remaining vertices, while constraint (21) specifies the upper bound on the path length. Moreover, constraint (22) guarantees that each POI belongs to only one category, whereas constraint (23) determines the maximum number of POIs for each category allowed in a path.

As a result, the main goal of integrating OP with social sensing is to utilize the *similarity matrix* previously obtained (subsection 4.4) to improve the quality of itineraries. Differently from [34], in this paper we exploit the algorithm for solving the orienteering problem developed in [5] rather than the one in [6], as it is an approximation algorithm with guaranteed bounds for the quality of the solution and as an anytime algorithm it provides a solution right from the start, which is then gradually improved.

Since POIs can be visited only once, Bolzoni et al. [5] use sets as route representatives. Thus, each search space node comprises a set of POIs, the related scores, and a potential extra score. Each element of the search space, then, represents a solution able to visit all the POIs in its set collecting their scores. The potential extra score measures how promising a search space node is. In particular, the algorithm computes the shortest path visiting all the POIs of a search space node, with (p_l, d) being the final arc leading to the destination. We consider the remaining budget time the remaining time at p_l. The potential extra score is an upper bound of the score we can get by visiting additional unvisited POIs between p_l and d. A trivial upper bound is to add up all the POIs p so that $c(p_l, p) + c(p) + c(p, d) - c(p_l, d)$ still allows us to arrive at d in time. However, with categories it is possible to have a stricter bound, since we consider only the top scoring POIs of each category satisfying the category constraints.

The algorithm then uses a double-ended priority queue (aka priority deque) that keeps the solution representatives in descending order of maximum potential score. In more detail, routes in the deque are represented as tuples comprising $\langle c, score, extra, \check{P} \rangle$ where $\check{P}$ is the set of POIs of the representative, c is the cost of passing through and visiting all the POIs starting from s and arriving at d, $score$ is the sum of the scores of the POIs in $\check{P}$, and $extra$ is an upper bound on additional scores that are still achievable. The *pot* (potential) of a representative is $score + extra$.

For instance, let us assume the initial representative $\check{P} = \varnothing$ for an empty route from s to d, then *cost* is the distance between s and d, $score$ is zero, and $extra$ contains the sum of the top scoring reachable POIs of each category. Assuming the category constraints are 2 *museums* and 1 *restaurant*, then $extra$ contains the scores of the top restaurant and the two top museums still in reach. The more POIs we have in $\check{P}$, the higher the *cost*, and the tighter the category constraint will be, allowing $extra$ to be more precise. If $extra$ is zero, it means it is not possible to visit any more POIs without violating a constraint. Usually, this upper bound is not a supremum because visiting some of the top scoring POIs will make it impossible to still reach others.

The priority deque keeps the representatives sorted by their potential score *pot* and in each step the algorithm removes the one with maximum *pot*, represented by $\langle c, score, extra, \check{P} \rangle$. Based on this removed route, we generate new representatives $\langle c_i, score_i, extra_i, \check{P} \cup p_i \rangle$ where p_i is a new POI (not in $\check{P}$), updating all its components appropriately. The new representatives are put back into the priority deque only if the sum of $score$ and $extra$ is higher than the score of the currently best known route.

Each time a representative is computed, the relative route is compared with the best known route. If it is better, it becomes the new best known. A route is better if it has a higher score, or given the same score has a lower cost. Theoretically, this algorithm would compute the optimal solution if we run it without any additional constraints. However, as OP is an NP-hard problem, we want to keep the runtime and memory usage manageable, so we employ it as an

approximation algorithm. To keep memory under control it is possible to limit the deque length: if, after inserting a new representative, the deque is longer than the limit, the minimum element (the one with the lowest potential score) is dropped.

As already mentioned, we only put a representative into the deque if its potential score *pot* is higher than the best known route score, as such a partial solution cannot possibly be expanded to an optimal solution. However, it is also possible to prune more aggressively by ignoring representatives whose *pot* is smaller than the best solution found so far multiplied by a *cut factor* t. In other words, *best* being the score of the best known solution, a representative is only enqueued if $pot > t \cdot best$. In this way we may lose the optimal solution, but since *extra* is an overestimation we can compute a provable bound (see [5] for a formal proof).

Limiting the length of the queue, we have to keep track of the highest potential we ignored, let it be max; then the optimal solution cannot be better than $max/best$ times the score of the returned solution. When using a cutting factor, t is the worst ratio between the scores of the optimal solution and the returned one.

The approach is an anytime algorithm, since it continuously generates and improves solutions. Thus, it is possible to keep runtime under control by returning the current best known solution after reaching a timeout.

While this is an efficient approximation algorithm with provable bounds, so far Bolzoni et al. assume that all the scores of the POIs are given as an input parameter [6, 5]. Determining and custom-tailoring the scores of POIs for individual users was left as an open problem. Here we solve this problem by combining an efficient algorithm for solving the OP with categories [5] with effective social sensing techniques, allowing us to automatically extract user preferences and map them to POIs scores. More formally, given a set U of users and a set P of POIs, the score of POI $p_j \in P$ for user $u_i \in U$ is calculated as follows:

$$score(u_i, p_j) = SIM_{ij} \tag{24}$$

5. Experimental Evaluation

We implemented the overall pipeline described in Figure 2 and evaluated the *accuracy* of the user interests in the POIs, the *user satisfaction* with reference to the suggested paths, and *run-time performance* of both the *social sensing* approach and the *orienteering* algorithm. In the following we provide some details on the experimental setting before discussing the results.

5.1. Environment

We ran the experiments on a machine with a 2.30GHz Intel Core i5 processor (3Mb of cache) with 8 GBytes of RAM running Archlinux, kernel version 4.9.6.

5.2. Data Sets

In order to evaluate the viability of our approach, we needed to construct data sets of both *user tweets* and *POI descriptions*, which we describe in the following.

5.2.1. OSN Users

Regarding the collected data set of OSN users, we downloaded the tweets of 549 users (on average 3200 tweets each) who are followers of the Municipality of Bolzano Twitter account[3]. Then, we randomly selected 20 users from the users who posted at least 100 tweets and who posted at least 60% of their tweets in Italian.

All the user tweets were then preprocessed as described in Section 4.2, before applying the schemes described in Section 4.

[3]These users have not put restrictions in their privacy settings, allowing us to do so.

5.2.2. POI Descriptions

In order to compare twitter accounts with POIs, we need to have a textual representation of POIs. This dataset was constructed manually by searching for textual descriptions of POIs in Bolzano that are interesting to a tourist. The POI documents were then processed according to the schemes in Section 4. Table 2 lists the set of considered POIs.

5.3. Results

First, we evaluate the *accuracy* of the social sensing approach ($RANK$ matrix) measured against a ground truth provided by human annotators. The annotators were informed about a subset of the tweets of a subset of users. They were asked to define a *top-k list* of POIs for each selected user based on their tweets. Second, we measured the *user satisfaction* of each selected Twitter user whom we suggested several tourist paths. Eventually, we also measured the *processing time* of both the social sensing approaches and the *orienteering algorithm*, illustrating how well this processing scales by using different numbers of tweets and various time budgets.

5.3.1. Evaluating Accuracy

Given the set of users $U = \{u_i\}_{i\in[1,m]}$ (where $m \in \mathbb{N}$ is the total number of users), the set of POIs $P = \{p_j\}_{j\in[1,n]}$ (where $n \in \mathbb{N}$ is the total number of POIs), and the set of tweets $T = \{\lambda_i \in \Lambda | \lambda_i.s = \text{“}e\text{”}\}$ (Table 1), let $U' = \{u'_i\}_{i\in[1,n_u]} \subset U$, $n_u \in \mathbb{N}$, $n_u < m$, $P' = \{p'_j\}_{j\in[1,n_p]} \subset P$, $n_p \in \mathbb{N}$, $n_p < n$, and $T' = \{t'_l\}_{l\in[1,n_t]} \subset T$, $n_t \in \mathbb{N}$, $n < |T|$ be the number of users, the number of POIs, and the number of tweets per user that n_{ann} annotators take into account while performing the rankings, respectively.

Additionally, let k be the number of POIs for which both the system and the annotators compute the rankings. More specifically, for each user $u \in U'$, let $R^a_u = \{r^a_i\}_{i\in[1,k]}$ denote the ranks computed by the algorithm, which are the POI IDs of the u-th row of the $RANK$ matrix (R_{u*}); on the other hand, let

Table 2: Set P' of POIs

ID	Name of POIs
2000	South Tyrol Museum of Archaeology - Bolzano
2001	Bolzano cathedral - Bolzano
2002	Earth Pyramids Renon - Soprabolzano
2003	Neptune's Fountain - Bolzano
2004	Argentieri Road - Bolzano
2005	Chiesa dei Domenicani - Bolzano
2006	Messner Mountain Museum Firmian - Bolzano
2007	Franciscan Friary - Bolzano
2008	Museo di scienze naturali dell'Alto Adige - Bolzano
2009	Museion Museum of Modern and Contemporary Art - Bolzano
2010	Arcades - Bolzano
2011	Piazza delle Erbe - Bolzano
2012	Bolzano Station - Bolzano
3000	Renon cableway - Bolzano
3001	Runkelstein Castle - Bolzano
3002	Twenty Shopping Mall - Bolzano
3003	Victory Monument - Bolzano
3004	Maretsch Castle - Bolzano
3005	Emberfly Art Gallery - Bolzano
3006	Teatro comunale bolzano - Bolzano
3007	Palazzo Menz - Bolzano
3008	Antico Municipio - Bolzano
3009	Teutonic Order Church - Bolzano
3010	Monument to King Laurin - Bolzano
3011	Bottai Road - Bolzano
3012	Civic Museum - Bolzano

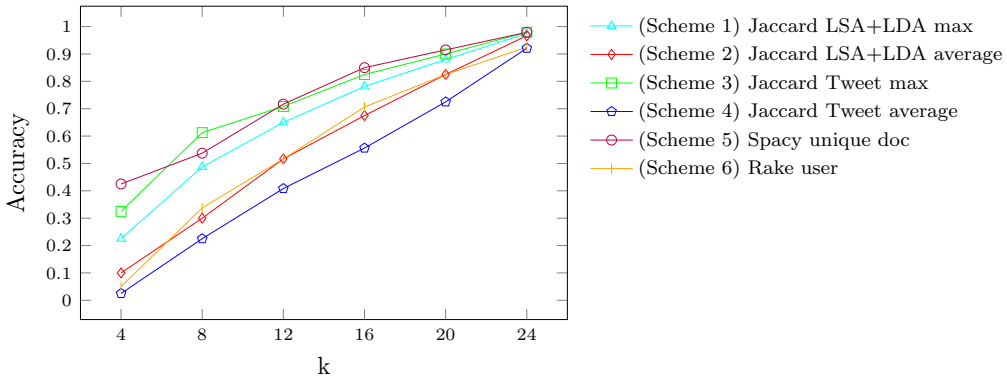


Figure 7: Accuracy results for our six schemes

$R_u^h = \{r_j^h\}_{j\in[1,k]}$ denote the ranks extracted by a human annotator. Then, for *each user* and *annotator*, the accuracy A is computed as follows:

$$A = \frac{|R_u^a \cap R_u^h|}{k}, \forall u \in U' \tag{25}$$

Clearly, A coincides with the classical definitions of both *precision* and *recall*, since both the system and the annotators are asked for *top-k lists* of POIs.

More specifically, in this use case we set $n_u = 20$, $n_p = 26$, $n_t = 3200$, and $n_{ann} = 3$. Table 2 shows the set P' of considered POIs.

Figure 7 shows the behavior of the average accuracy when varying k. More specifically, we reported the accuracy achieved by the six schemes which are *Jaccard LSA+LDA max, Jaccard LSA+LDA average, Jaccard Tweet max, Jaccard Tweet average, spaCy, and RAKE*. From Figure 7, we can infer that all the listed approaches have a similar linear increasing trend; *spaCy* and *Jaccard Tweet max* achieve better accuracy than the rest of the schemes. When k is relatively small ($k = 4$), *spaCy* outperforms *Jaccard Tweet max*. For $k = 8$, *Jaccard Tweet max* shows the best accuracy. Further increasing k makes the accuracy scores of *spaCy* and *Jaccard Tweet max* converge and nearly overlap. Between $k = 12$ and $k = 20$, there is no significant change in terms of the linear tendency and the relative positioning of the different schemes. As we will see,

the main advantage of *Jaccard Tweet* is that it can be significantly faster than *spaCy*.

5.3.2. Evaluating User Satisfaction and Route Recall

In order to further evaluate the results, we have conducted a user study to evaluate the *user satisfaction* and the *route recall* for the suggested routes. This user study intends to evaluate (1) if the component of social sensing will increase user satisfaction, (2) which of the six proposed score schemes leads to a higher user satisfaction, and (3) the percentage of POIs in each route proposed to the user that are actually listed in his/her top-k POI collection (i.e., computing the *route recall*). Therefore, we derived nine customized routes for each user. Six of them are based on the six score schemes, one is using fixed scores for each POI, and, as baselines, the second to last one assigns scores randomly to the POIs, whereas the last one just picks from the user ranking the top-k POIs which are visitable within the time constraints.

We used active twitter accounts for sensing the user preference and POIs from the city of Bolzano in Italy. As Bolzano is a touristic city and most POIs are within walking distance, a user does not need to concern themselves with any transportation issues. Also, the users in our study were not aware of the different scoring schemes and also did not know that we analyzed their tweets.

We presented the following scenario to the user: assume you would like to visit Bolzano and every POI there is new to you. Given a fixed total visiting time of three hours and twenty minutes, you are not able to visit all the POIs. We prepared a total of **nine** suggested routes. As an example, Table 4 shows the routes proposed to the user *X*. Each route contains a set of POIs with an orienteering route such as shown in Figure 8. The transportation means is walking. After reviewing all the proposed routes and related POIs, we asked users to rank the nine proposed routes based on their own preferences, thus allowing the computation of the *user satisfaction*.

The generated routes were computed using the reference implementation of the best first search algorithm in Section 4.6. Each pair of POI distances is

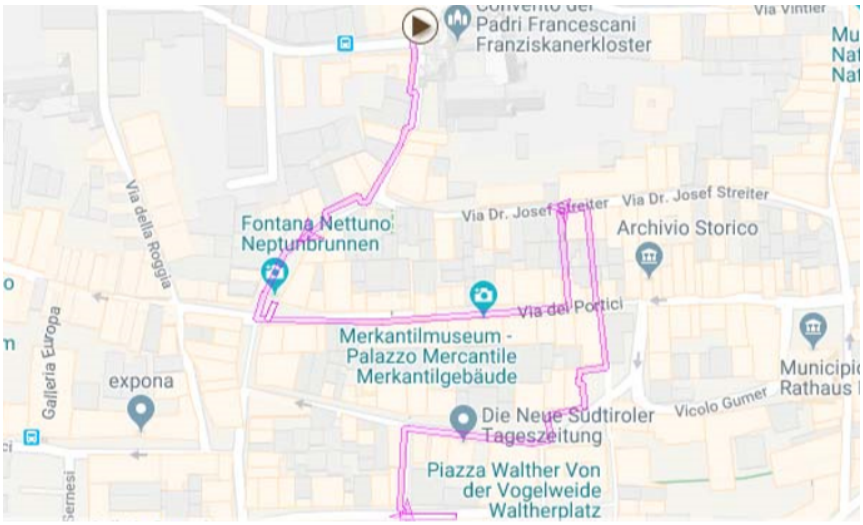


Figure 8: Route presented to the user

computed as the walking time distance according to Open Street Map and the POI scores are provided by social sensing. That leaves the budget time, starting, and ending points. We executed experiments for tours from 30 minutes up to 200 minutes (3 hours and 20 minutes), here we focus on the results for the tours lasting 200 minutes. Furthermore, we assumed that to actually collect the score of a POI, a tourist needs to stay for 10 minutes at a POI. The tour starts and ends near the Franciscan Friary, which is a well-known and fairly central POI in Bolzano.

The 10 minutes visiting time is meant to be a bare minimum needed to actually get something out of a POI, but still long enough to make an impact on the budget time. The algorithm ensures that the destination is reached in less than the budget time, but in reality we expect the tours to last longer, as people will take pauses and stay in POIs longer than 10 minutes, yet even a 200 minute tour is doable in an afternoon.

Thus, in order to compute the *route recall*, we first asked each user u to explicitly provide a ranked list of POIs $R_u^{gt} = \{r_i^{gt}\}_{i \in [1,n]}$, where gt stands for

ground truth. Then, assuming $P_u^a = \{p_i^a\}_{i\in[1,k]}$ to be the route (path) computed by the algorithm a for user u, and passing through the k-sequence of POIs $\{p_i^a\}_{i=1}^k$, the route recall for user u and path P_u^a is computed as follows:

$$R_{P_u^a} = \frac{|R_u^{gt} \cap P_u^a|}{k} \tag{26}$$

Thus, the *route recall* denotes the percentage of POIs in each route of length k matching the user top-k list of POIs.

For instance, Table 4 shows the routes proposed to the user X as well as the related *route recall*, that we computed by comparing the routes against the POI ranking list of user X shown in Table 3.

As a result, we can infer that for the user X, the route which achieves the highest recall is the *(Scheme 5) Spacy unique doc* (63.64%). In this case study, also the *random path* accidentally achieves a good accuracy, but this is clearly not valid in the general, average, case.

Additionally, the last row of Table 4 shows a path which just covers the top-k POIs of the list R_u^{gt} of the user u , where k is the maximum number of POIs visitable within the time constraints. Clearly, its route recall is 100%, but evidently its number of visited POIs is much lower with respect to the ones passed through by the other routes, since the path generation is not optimized in this case.

However, considering all the users, the results of the user study is that five out of six routes with social sensing are ranked higher than the three routes not exploiting social sensing, and also achieve a higher *route recall* (excluding the *Ideal Route*). This means, that adding social sensing to trip planning tends to have a positive impact on both user satisfaction and route recall. The five higher ranked routes are using the following schemes: spaCy unique doc, Jaccard LSA+LDA max, RAKE, Jaccard Tweet average, and Jaccard Tweet Max. The top three ranked routes are the variants spaCy unique doc (scheme 5), Jaccard LSA+LDA max (scheme 1), and RAKE (scheme 6), in that order. This is a strong indication that these three schemes are the top contenders when it

Table 3: POI Ranking List of User X

Rank	**ID**	**Name of POIs**
1	2011	Piazza delle Erbe - Bolzano
2	3000	Renon cableway - Bolzano
3	2010	Arcades - Bolzano
4	2007	Franciscan Friary - Bolzano
5	2001	Bolzano cathedral - Bolzano
6	3001	Runkelstein Castle - Bolzano
7	3004	Maretsch Castle - Bolzano
8	2005	Chiesa dei Domenicani - Bolzano
9	2000	South Tyrol Museum of Archaeology - Bolzano
10	2009	Museion Museum of Modern and Contemporary Art - Bolzano
11	2002	Earth Pyramids Renon - Soprabolzano
12	3008	Antico Municipio - Bolzano
13	3006	Teatro comunale bolzano - Bolzano
14	3011	Bottai Road - Bolzano
15	3007	Palazzo Menz - Bolzano
16	2004	Argentieri Road - Bolzano
17	3005	Emberfly Art Gallery - Bolzano
18	2003	Neptune's Fountain - Bolzano
19	2008	Museo di scienze naturali dell'Alto Adige - Bolzano
20	3012	Civic Museum - Bolzano
21	3010	Monument to King Laurin - Bolzano
22	3009	Teutonic Order Church - Bolzano
23	2006	Messner Mountain Museum Firmian - Bolzano
24	3003	Victory Monument - Bolzano
25	2012	Bolzano Station - Bolzano
26	3002	Twenty Shopping Mall - Bolzano

Table 4: The nine routes suggested to the user X

Route	**POIs with Start and End 2007**	**Route Recall**
(Scheme 1) Jaccard LSA+LDA max	⟨2008, 3000, 2012, 3007, 2005, 2009, 3012, 3003, 2000⟩	50%
(Scheme 2) Jaccard LSA+LDA average	⟨2010, 3007, 2009, 3005, 3003, 3012, 2011, 2005, 2012, 3009⟩	45.45%
(Scheme 3) Jaccard Tweet max	⟨2010, 3007, 3006, 2009, 3005, 3003, 3012, 2011, 2005, 2008⟩	45, 45%
(Scheme 4) Jaccard Tweet average	⟨2003, 3012, 2000, 2009, 2010, 2004, 2011, 2005, 2012, 3010⟩	54.54%
(Scheme 5) Spacy unique doc	⟨2011, 2005, 3006, 2001, 3007, 3008, 2010, 2000, 3012, 2009⟩	63.64%
(Scheme 6) Rake user	⟨2011, 2003, 2005, 2001, 3007, 3008, 2000, 3005, 3012⟩	50%
Fixed 500	⟨2011, 2003, 2010, 3008, 2004, 3007, 2005, 2001, 2008, 3011⟩	45.45%
Random	⟨2011, 2000, 3012, 2005, 2001, 3007, 2003, 3008, 3010, 3000⟩	54.54%
Ideal Route (Baseline)	⟨2011, 3000, 2010, 2007, 2001, 3001⟩	100%

comes to increasing the user satisfaction. The only social sensing scheme that is not able to outperform the Fixed 500, Random, and Ideal Route approaches is Jaccard LSA+LDA average. One of the reasons may be that the average of all the Jaccard values introduces more topics that are not really favored by users.

We also conducted an interview with users to explore the reasons why certain routes are preferred. We found that if a POI that is strongly preferred by a user is missing in a proposed tour, the user will rank the route lower. Also, if some non-preferred POIs are on the path to interesting POIs, the user will also consider to rank that route lower. As this is a walking tour, the environ-

ment plays a role as well: walking along a river or in a pedestrian zone usually increases the user satisfaction. Additionally, the Ideal Routes are ranked lower than the routes computed with the support of social sensing, since users prefer to visit as many POIs as possible within the time constraints and, even more important, do not want to follow paths which are not optimized in terms of total traveling time. Finally, we also report one of the evaluations of the approach in the field by one volunteer user who tested our framework for seven days in Bolzano. The feedback we received from this user was positive, which promises to make our approach applicable for real-world use.

5.3.3. Evaluating Performance

In this section, we evaluate the performance of the framework in terms of running times used by both the *social sensing* approach and the *orienteering* algorithm. Figure 9 shows how the six social sensing approaches scale when varying the number of analyzed tweets (each of the values shown on the x-axis of Figure 9 is the number of tweets of a specific user). In principle all of them achieve reasonable running times (times are in the order of seconds even when processing more than 2000 tweets), although the approaches shown in schemes 1 and 2 are much slower than all the others. The fastest one is definitely the approach based on *spaCy* (Scheme 5), which can be even used to monitor and process large numbers of Tweets in real time.

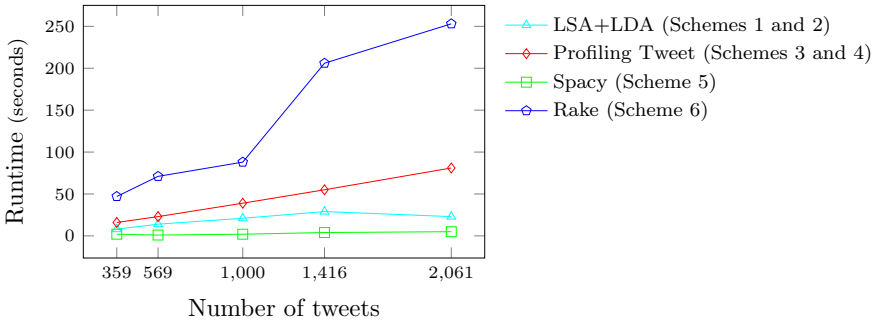
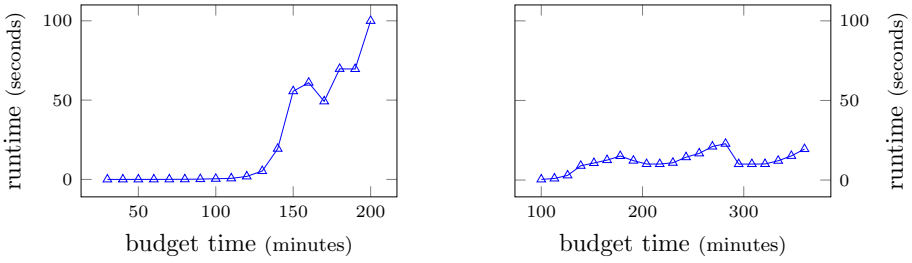


Figure 9: Run Time Performance - Social Sensing

Additionally, in the following we discuss the performance of the *orienteering algorithm* in terms of running time. We did not apply aggressive pruning in the form of a cut factor, but we limited the run time to one minute. Figure 10(a) shows how the run time of the algorithm scales for different time budgets, whereas Figure 11 refers to the run time of the orienteering algorithm with that of the social sensing approaches. Even though we limited the run time to one minute, the orienteering algorithm can go beyond that, since it only stops at the end of a complete expansion phase.

It is also interesting to note that the run time does not grow monotonically with the budget time. The reason for this is that the POIs are not distributed evenly throughout the city, which means that for different budgets different sets of POIs can be reached, making it more (or less) difficult to maximize the overall score of a route.



(a) run time limit 60s (b) run time limit 10s

Figure 10: Route generation time, increasing the time budget.

If a fast response time is important, we can also lower the run time limit of the orienteering algorithm. Figure 10(b) shows how the algorithm scales for different budgets with a run time limit of ten seconds (up to routes with a length of 360 minutes). Reducing the response time to ten seconds only has a minor impact on the quality of the generated routes in terms of the overall score. On average, the scores of the routes produced by a ten-second execution were only about 1% lower than those generated by running the orienteering algorithm for one minute.

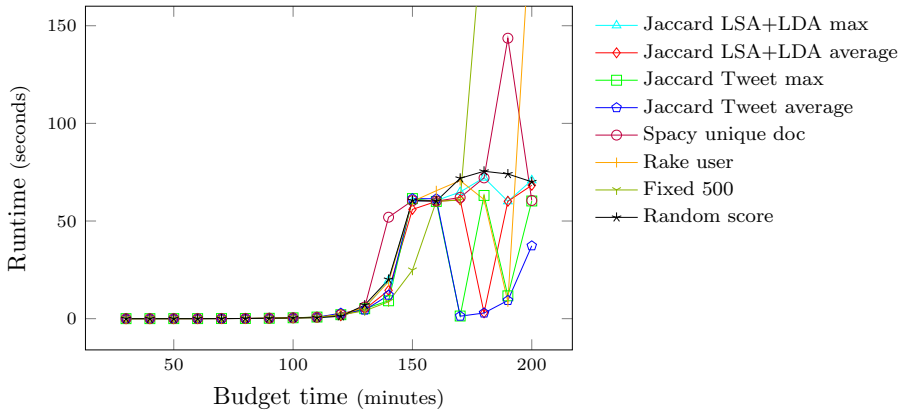


Figure 11: Runtime of each score

5.3.4. Final Remarks

Overall, we were able to effectively integrate all of the six proposed approaches for social sensing with the algorithm for solving the orienteering problem with category constraints, and achieve good performance in terms of *accuracy*, *user satisfaction*, and *efficiency*; in fact, five out of the six approaches are ranked by the users better than the orienteering algorithm not exploiting social sensing. However, it turns out that the approach which performs best is the one based on *spaCy* (scheme 5).

Furthermore, in personalized trip planning, users may expect to explore new or surprising travel experiences that usually cannot be captured by stable user preferences [42]. In this paper, we only consider and evaluate route quality as the alignment between the personalized POIs and user preferences. Other evaluation factors such as serendipity or coverage [43] of POIs are out of the scope of this paper. Since the main goal of the tourist trip planning is to recommend personalized routes for visiting highly preferred POIs, we provided each user with eight possible routes to choose from (six of which derived the user preferences by social sensing).

6. Conclusions

One widely used preference elicitation technique is to explicitly ask if a user likes or dislikes a certain item that was previously experienced by the user. In recommender systems, the preference for an item is usually represented by a rating for the item [44]. Although from the user's perspective explicit preference elicitation can be seen as a redundant procedure with considerable effort on the side of the user, it is still employed due to its effectiveness in many application domains [45]. However, there is always the risk of users losing interest in a system because of the effort they have to invest to communicate their preferences. Thus, we advocate the use of implicit preference elicitation whenever possible (i.e., whenever the quality of a service does not suffer) to keep the overhead as small as possible. The advent of social sensing techniques provides us with powerful tools for automating preference elicitation.

More concretely, we propose an innovative method for customizing the solutions to orienteering problems by applying six different schemes for social sensing. In tourist trip planning, we identify user interests by sensing their Twitter profile and are able to create an ordered list of POIs corresponding to these interests. In this way, an orienteering algorithm can be custom-tailored to a user by considering the semantic closeness between POIs and a user's social contents.

We implemented a prototype and evaluated its accuracy, user satisfaction levels, and computational performance. The evaluation has shown the effectiveness of exploiting social sensing for improving tourist trip planning. In particular, the approach that has shown the best user satisfaction is the one based on the *spaCy* tool. Thus, the evaluation demonstrates that it is very efficient and promising to integrate social sensing with orienteering-based tourist trip planning, making it applicable to real-world use cases.

In the future we want to build on this work by making it scale to even larger numbers of users and POIs, and we want to extend our approach to other languages and additional social networks, such as Facebook or Instagram. We

do not see switching to another Indo-European language as an issue, since, for instance, the fastText framework [32] is available for 157 different languages.

References

[1] S. Liu, L. Wang, A self-adaptive point-of-interest recommendation algorithm based on a multi-order markov model, Future Generation Computer Systems 89 (2018) 506–514. doi:10.1016/j.future.2018.07.008.

[2] M. Eirinaki, J. Gao, I. Varlamis, K. Tserpes, Recommender systems for large-scale social networks: A review of challenges and solutions, Future Generation Computer Systems 78 (2018) 413–418. doi:10.1016/j.future.2017.09.015.

[3] S. Milovanovi, Z. Bogdanovi, A. Labus, D. Bara, M. Despotovi-Zraki, An approach to identify user preferences based on social network analysis, Future Generation Computer Systems 93 (2018) 121 – 129. doi:10.1016/j.future.2018.10.028.

[4] S. Rose, D. Engel, N. Cramer, W. Cowley, Automatic keyword extraction from individual documents, in: M. W. Berry, J. Kogan (Eds.), Text Mining. Applications and Theory, John Wiley and Sons, Ltd, 2010, pp. 1–20. doi:10.1002/9780470689646.ch1.

[5] P. Bolzoni, S. Helmer, Hybrid best-first greedy search for orienteering with category constraints, 2017, pp. 24–42. doi:10.1007/978-3-319-64367-0_2.

[6] P. Bolzoni, F. Persia, S. Helmer, Itinerary planning with category constraints using a probabilistic approach, in: D. Benslimane, E. Damiani, W. I. Grosky, A. Hameurlain, A. Sheth, R. R. Wagner (Eds.), Database and Expert Systems Applications, Springer International Publishing, Cham, 2017, pp. 363–377. doi:10.1007/978-3-319-64471-4_29.

[7] T. Tsiligirides, Heuristic methods applied to orienteering, The Journal of the Operational Research Society 35 (9) (1984) 797–809. doi:10.1057/jors.1984.162.

[8] E. H. Lu, C. Lin, V. S. Tseng, Trip-mine: An efficient trip planning approach with travel time constraints, in: 2011 IEEE 12th International Conference on Mobile Data Management, Vol. 1, 2011, pp. 152–161. doi:10.1109/MDM.2011.13.

[9] G. Righini, M. Salani, Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming, Computers & Operations Research 36 (4) (2009) 1191 – 1203. doi:https://doi.org/10.1016/j.cor.2008.01.003.

[10] C. Keller, Algorithms to solve the orienteering problem: A comparison, European Journal of Operational Research 41 (2) (1989) 224 – 231. doi:https://doi.org/10.1016/0377-2217(89)90388-3.

[11] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, M. Minkoff, Approximation algorithms for orienteering and discounted-reward tsp, in: 44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings., 2003, pp. 46–55. doi:10.1109/SFCS.2003.1238180.

[12] C. Chekuri, N. Korula, M. Pál, Improved algorithms for orienteering and related problems, ACM Trans. Algorithms 8 (3) (2012) 23:1–23:27. doi:10.1145/2229163.2229167.

[13] Z. Sevkli, F. E. Sevilgen, Variable neighborhood search for the orienteering problem, in: A. Levi, E. Savaş, H. Yenigün, S. Balcısoy, Y. Saygın (Eds.), Computer and Information Sciences – ISCIS 2006, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 134–143. doi:10.1007/11902140_16.

[14] M. F. Tasgetiren, A. E. Smith, A genetic algorithm for the orienteering problem, in: Proceedings of the 2000 Congress on Evolutionary Com-

putation. CEC00 (Cat. No.00TH8512), Vol. 2, 2000, pp. 910–915 vol.2. doi:10.1109/CEC.2000.870739.

[15] Q. Wang, X. Sun, B. L. Golden, J. Jia, Using artificial neural networks to solve the orienteering problem, Annals of Operations Research 61 (1) (1995) 111–120. doi:10.1007/BF02098284.

[16] P. Bolzoni, S. Helmer, K. Wellenzohn, J. Gamper, P. Andritsos, Efficient itinerary planning with category constraints, in: 23nd ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (ACM SIGSPATIAL'14), 2014, pp. 203–212. doi:10.1145/2666310.2666411.

[17] W. Huang, I. Weber, S. Vieweg, Inferring nationalities of twitter users and studying inter-national linking, in: Proceedings of the 25th ACM Conference on Hypertext and Social Media, HT '14, ACM, New York, NY, USA, 2014, pp. 237–242. doi:10.1145/2631775.2631825.

[18] J. Bentwood, Distributed influence: Quantifying the impact of social media, Edelman White Paper (2007).

[19] R. Tinati, L. Carr, W. Hall, J. Bentwood, Identifying communicator roles in twitter, in: Proceedings of the 21st International Conference on World Wide Web, WWW '12 Companion, ACM, New York, NY, USA, 2012, pp. 1161–1168. doi:10.1145/2187980.2188256.

[20] E. D'Avanzo, G. Pilato, M. Lytras, Using twitter sentiment and emotions analysis of google trends for decisions making, Program 51 (3) (2017) 322–350. arXiv:https://doi.org/10.1108/PROG-02-2016-0015, doi:10.1108/PROG-02-2016-0015.

[21] S. Gianvecchio, M. Xie, Z. Wu, H. Wang, Measurement and classification of humans and bots in internet chat, in: Proceedings of the 17th Conference on Security Symposium, SS'08, USENIX Association, Berkeley, CA, USA, 2008, pp. 155–169.
URL http://dl.acm.org/citation.cfm?id=1496711.1496722

[22] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, WWW '01, ACM, New York, NY, USA, 2001, pp. 285–295. doi:10.1145/371920.372071.

[23] T. K. Landauer, P. W. Foltz, D. Laham, An introduction to latent semantic analysis, Discourse processes 25 (1998) 259 – 284. doi:10.1080/01638539809545028.

[24] T. K. Landauer, S. T. Dumais, A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge, Psychological review 104(2) (1997) 211 – 223. doi:10.1037//0033-295X.104.2.211.

[25] F. Agostaro, A. Augello, G. Pilato, G. Vassallo, S. Gaglio, A conversational agent based on a conceptual interpretation of a data driven semantic space, Lecture Notes in Artificial Intelligence 3673(2) (2005) 381 392. doi:10.1007/11558590_39.

[26] S. Santilli, L. Nota, G. Pilato, The use of latent semantic analysis in the positive psychology: A comparison with twitter posts, Semantic Computing (ICSC), 2017 IEEE 11th International Conference on IEEE (2017) (2017) 494498. doi:10.1109/ICSC.2017.60.

[27] D. Blei, A. Ng, M. Jordan, Latent dirichlet allocation, Journal of Machine Learning Research 3 (January 2003) 9931022.

[28] T. Hofmann, Probabilistic latent semantic indexing, in: Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99, ACM, New York, NY, USA, 1999, pp. 50–57. doi:10.1145/312624.312649.

[29] D. M. Blei, Probabilistic topic models, Commun. ACM 55 (4) (2012) 77–84.

[30] W. Darling, A theoretical and practical implementation tutorial on topic modeling and gibbs sampling, Proceedings of the 49th annual meeting of

the association for computational linguistics: Human language technologies (Dec 2011) 642–647.

[31] Y. Whye Teh, D. Newman, M. Welling, A collapsed variational bayesian inference algorithm for latent dirichlet allocation, Proceedings of NIPS 6 (2006) 1378–1385.

[32] fasttext : Library for efficient text classification and representation learning (2018).
URL https://fasttext.cc/

[33] spacy, the industrial-strength natural language processing (2018).
URL https://spacy.io/

[34] F. Persia, S. Helmer, S. Pugacs, G. Pilato, Social sensing for improving the user experience in orienteering, in: 2019 IEEE 13th International Conference on Semantic Computing (ICSC), 2019. doi:10.1109/ICOSC.2019.8665498.

[35] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.

[36] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, T. Mikolov, Learning word vectors for 157 languages, arXiv preprint arXiv:1802.06893.

[37] F. Persia, S. Helmer, A framework for high-level event detection in a social network context via an extension of iseql, in: 2018 IEEE 12th International Conference on Semantic Computing (ICSC), 2018, pp. 140–147. doi:10.1109/ICSC.2018.00028.

[38] F. Amato, A. Santo, V. Moscato, F. Persia, A. Picariello, Detecting unexplained human behaviors in social networks, in: 2014 IEEE International Conference on Semantic Computing, 2014, pp. 143–150. doi:10.1109/ICSC.2014.21.

[39] F. Amato, A. Castiglione, A. D. Santo, V. Moscato, A. Picariello, F. Persia, G. Sperl, Recognizing human behaviours in online social networks, Computers & Security 74 (2018) 355 – 370. doi:https://doi.org/10.1016/j.cose.2017.06.002.

[40] G. Pilato, G. Vassallo, Tsvd as a statistical estimator in the latent semantic analysis paradigm, IEEE Transactions on Emerging Topics in Computing 3 (2) (2015) 185–192. doi:10.1109/TETC.2014.2385594.

[41] A. Cuzzocrea, G. Pilato, Taxonomy-based detection of user emotions for advanced artificial intelligent applications, Hybrid Artificial Intelligent Systems, Lecture Notes in Computer Science 10870 (2018) 573 – 585. doi:10.1007/978-3-319-92639-1_48.

[42] J. Xu, J. Chen, R. Zhou, J. Fang, C. Liu, On workflow aware location-based service composition for personal trip planning, Future Generation Computer Systems 98 (2019) 274 – 285. doi:10.1016/j.future.2019.03.010.

[43] M. Ge, C. Delgado-Battenfeld, D. Jannach, Beyond accuracy: evaluating recommender systems by coverage and serendipity, in: Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010, 2010, pp. 257–260. doi:10.1145/1864708.1864761.

[44] R. Logesh, V. Subramaniyaswamy, V. Vijayakumar, X.-Z. Gao, V. Indragandhi, A hybrid quantum-induced swarm intelligence clustering for the urban trip recommendation in smart city, Future Generation Computer Systems 83 (2018) 653 – 673. doi:10.1016/j.future.2017.08.060.

[45] A. Pommeranz, J. Broekens, P. Wiggers, W.-P. Brinkman, C. M. Jonker, Designing interfaces for explicit preference elicitation: a user-centered investigation of preference representation and elicitation process, User Modeling and User-Adapted Interaction 22 (4) (2012) 357–397. doi:10.1007/s11257-011-9116-6.