Automatic generation of human-like route descriptions: a corpus-driven approach

Rafael Teles[a], Bruno Barroso[a], Adolfo Guimaraes[b], Hendrik Macedo[a]
[a] Computing Department, Federal University of Sergipe, Sergipe, Brazil
Email: rafast.telles@gmail.com, brunokreutz@hotmail.com, hendrik@ufs.br
[b] Computer Science Department, Federal University of Minas Gerais, Minas Gerais, Brazil
Email: adolfopg@dcc.ufmg.br

***Abstract*— Most of Web applications combines differents services, features and contents in order to enable the creation of new features and services. Such systems are called mashups. One of the most popular kind of mashups are the location ones that use geographic data to provide functionalites to users. The RotaCerta is a location system that uses the Google Maps and perform Natural Language Generation to provide textual descriptions of routes between two different locations. The great advantage of RotaCerta is the use of points of interest (POI) to describe routes. POIs help the user to understand and assimilate the route. However, RotaCerta suffers from a several limitation: the need for manualy updating of a POIs dataset. Such work is exhausting, costly and greatly limits their use. Another point to highlight is the poor linguistic variability of texts it provides. In this work, we propose a mechanism to enable automatic feeding of POIs and a corpus-driven approach to enhance the linguistic variability of location mashups such as RotaCerta. We adopt both manual and automatic generation of new textual templates. In order to assess the quality of the routes descriptions, we use TF-IDF and cosine distance to calculate the similarity between descriptions of routes created by human volunteers and descriptions generated by the proposed approach. Route generation examples have been performed for three different brazilian cities. We also show that the text generated from the new template base is more similar to the texts used by people when describing routes if compared to Google Maps.**

***Index Terms*— Mashups, Location Systems, Natural Language Generation, Points of Interests**

I. Introduction

THERE is a wide variety of mobile devices as mobile phones, smartphones, tablets and PDAs supporting alternative communication facilities such as WiFi, 3G and GPS. Many applications make use of these technologies in order to provide new services and features. One of these services are location systems, which are used to determine a user's position on a map and route between two geographic points. Google Maps[1] is one of the most popular location system. However, the routes descriptions are far from the way people use to communicate. By providing information about location of establishments, people often make use of landmarks/reference (Points of Interest - POI) to facilitate the assimilation of the route by the concerned person. The sentence "Go on The Street Y and after passing by the Pharmacy X, turn on the right.", for instance, seems to be more intuitive than something like "Go on The Street northbound for 500 meters and then turn right". Coordinates and cardinal notion of distance are far more difficult for human understanding.

Some approaches in the concerned literature aim to propose the inclusion of POIs on route suggestions in order to improve the description for the user. One of these systems is the GuiaMais[2]. Although it is quite similar to Google Maps in defining routes, it allows for the searching of POIs. The user provides the type of property he/she is looking for, such as pizzerias, bars, restaurants or shopping malls at a specific neighborhood, city or state. The system returns a map of the area with several POIs as requested. A great limitation of GuiaMais is that the POI information is not incorporated into the description of routes.

The system Already Talking Points [1] describes an urban orientation system based on reference points. The authors advocate the idea that a walking route may be better presented with the aid of POIs. In addition to include reference points in the descriptions of routes, the system provides an vocal-enabled interface. This allows its use primarily by the visually impaired. However, the feeding of POIs should be done manually, which requires enourmous effort and limits its usage.

Another system that uses POIs on its route suggestions are proposed by [2]. The author emphasizes the advantages of using POIs in the description of the routes, both to provide confirmation that the user is going in the right direction and also to facilitate the memorization of routes. The system implements an algorithm that weights references according to their characteristics in order to select the points that can be more easily identified by people. However, the basis of POIs only contains data regarding Australia.

RotaCerta [3] incorporates information of POIs on route suggestions. The system has its own built-in POIs and uses techniques from Natural Language Generation (NLG) along with the Google Maps API to describe routes in a way similar to that used by people on a day-to-day. Although its interesting results, the RotaCerta

[1] http://maps.google.com

[2] http://www.guiamais.com.br/

doi:10.4304/jetwi.5.4.413-423

suffers from problems of maintainability and updating the database of POIs. The feeding of POIs database is manually done, which considerably limits system scalability. Another limitation to highlight is the low linguistic variability of the text generated to describe the routes.

This paper extends the work of RotaCerta [3] along three different axis: (i) proposal of an automated mechanism to update the POI database, (ii) improve the NLG module in order to augment the linguistic variability of the set of pre-defined templates that constitute the description of the route and, finally, (iii) perform an assessment of the quality of the generated text. The goal is to ensure that the system can be used to generate more human-like route descriptions to any place of interest. Unlike the original work, our approach considers the integration with the Google Places[3] system, which consists of a collaborative system for adding POIs of several countries, and a new approach to associate POI to generated routes. The approach used to enhance linguistic variability is corpus-driven. A web page was created to allow the collection of route descriptions provided by volunteers. The extraction of new templates has been performed in two different ways: (i) manually and (ii) automatically, by the adaptation of the algorithm to generating paraphrases described in [4] .

The remainder of this paper is organized as follows. In Section II, we describe the background technologies used to the system development: Mashups, Natural Language Generation and Generation of Paraphrase-based data. Section III presents implementation details and highlight the differences to previous system. Performed experiments, results and discussions are shown in section IV. Finally, in Section V we conclude the work and discuss some possible extensions and future investigations.

II. Technological background

A. Mashups

The term Web 2.0 is commonly associated with Web applications that facilitate the sharing of information interactively such as forums, Web sites, social networks, blogs and mashups [5]. Although the concept of Web 2.0 suggest a new version of the World Wide Web, this does not refer to any change concerning technical specifications, but rather in the way that developers and end-users use the Web [6].

Originally, the term mashup was used to describe mixing or blending of two or more music tracks, commonly used by DJ's. In the context of Web, mashups are simply a new way of developing applications by combining services, features and content already available. Such elements can be formatted as RSS feeds, XML and its derivatives, HTML, Flash or any other type of graphics.

The mapping mashups are one of the most popular forms of mashups. Its main feature is the use of maps, usually used to determine routes between two points. Based on an initial geographical reference (longitude and latitude), these systems are able to describe a route to a target location. The result of this application is typically a graphical representation of the path and textual description. Currently, 81.82% of mapping mashups make use of the Google Maps API [7]. The availability of such an API [8] can be seen as a major factor for such indices [3].

Among the services provided by Google Maps API, can be highlighted: (1) the creation of paths based on source and destination addresses using the Google Directions, (2) calculate the distance between two points, (3) information of latitude and longitude, (4) coordinates of a given address through the Reverse Geocode, and (5) detailed information of a specific point such as street name, number, city, neighborhood, zip code and country.

B. Natural Language Generation

Natural Language Generation (NLG) is a field of Artificial Intelligence that addresses computational systems able to produce understandable texts in a particular human language, starting from non-linguistic data. NLG systems use knowledge about language and the application domain to automatically produce documents, reports, and other [9].

Figure 1 gives an example of a system for generating natural language descriptions of weather events from daily meteorological records [10]. Note that from daily records of temperature and rainfall it is possible to produce textual summarization of climatic events occurred in the month.

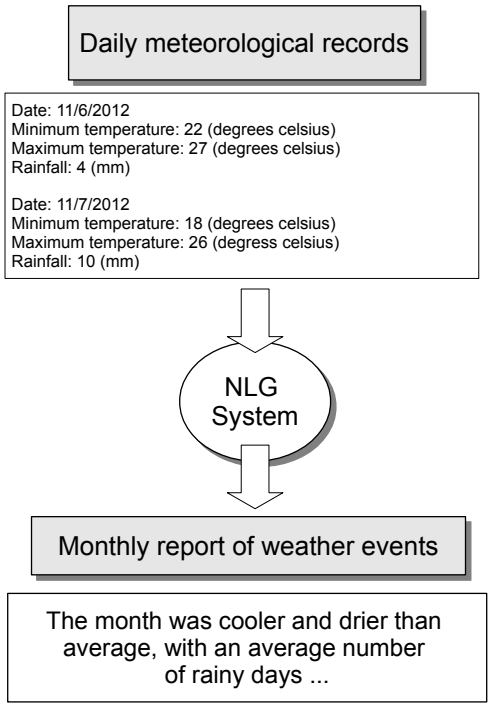

Figure 1. NLG system example

The final text of an NLG pipeline should meet the following features: (1) be linguistically correct, (2) clearly convey the information input, (3) respond to the proposal

[3]http://code.google.com/apis/maps/documentation/places/

for which it is being generated, and (4) appear fluent enough to avoid a mechanical communication [11]. Another aspect that can be taken into account is the generated text format. This can be raw text or include more sophisticated formatting elements to meet specific interface requirements, HTML, VXML, LaTeX, and others.

NLG systems is usually classified into 03 (three) categories (in decreasing order of complexity): (1) use of preprogrammed responses (canned text), (2) use of standardized sentences (template) and (3) phrase-based text generation. Each of these differ by their complexity and flexibility of results [11].

The basic idea behind canned text is to associate to each possible systems response at a given context, an answer in natural language. This method is satisfactory when it provides a small number of answers. As the number of system settings increases, this technique tends to become impracticable due to the huge amount of combinatory results [3].

NLG systems based on templates map the input data to a linguistic structure [12]. This linguistic structure may contain gaps or slots. The output result is obtained when all slots are filled or replaced by the language structures that do not have slots [13]. Suppose we want a system based on NLG templates be able to generate sentences to a train station. The description of such system can be seen in [12] and can start from a semantic representation informing that the train 306 leaves Aberdeen at 10:00 AM:

$$\text{Departure}(train_{306}, location_{abdn}, time_{1000}),$$

which is directly associated with a template such as:

[train] *is leaving* [town] *now*

The slots represented by [train] and [town] are filled by relevant information obtained from a table. This template is used only when the time said (10:00 AM) is close to the announcement message. Other templates should be used to create ads for the past or future.

Phrase-based text generation use an indirect mapping between input data and linguistic form. Such systems take as input a semantic representation of data. This is then subject of successive transformations until a linguistic structure is produced. Various system components can operate on this NLG defining, for instance, that 10:00 AM is the estimated time for the ad to be displayed and thereby transforming the input into an intermediate representation such as follows:

$$Leave_{present}(train_{demonstrative}, Aberdeen, now)$$

It is possible to note that the lexical items were determined when the linguistic morphology was still missing. This intermediate representation, in turn, can be transformed into an appropriate sentence such as:

This train is leaving Aberdeen now

The details of the sentence may vary. Such systems may contain several intermediate representations.

C. Paraphrase generation based on data

The paraphrase is usually considered a way for preserving meaning. If we are dealing with a essentially linguistic textual form, then we can say that the text A and B are paraphrases of each other, if both texts A and B have the same meaning [14]. In other words, the paraphrase is an alternative way of rewriting a text in the same language, without losing the semantic content of the original text.

Paraphrases may exist at the level of words, also known as lexical paraphrase, in which the most common are synonyms, for example, (car, automobile) and (dog, puppy). The hypernym is another example of lexical paraphrase. In this case, a word is more general or specific than the other, as in the examples (shoes, boots) and (fruit, orange).

Paraphrasing is also present at the level of sentences, as in the examples (I finished my work, I finished my task). At this level, it is possible to generate paraphrases by simply replacing one or more words in the original phrase by others with the same meaning.

The concept of distributional similarity, extremely popular technique used in the generation of paraphrase, states that words or phrases that share the same distribution - the same set of words in the same context in a set of texts written in a language that serves as a database for linguistic research (corpus) - tend to have similar meanings [15]. A commonly used model for calculating the similarity is the N-gram model.

The N-gram model aims to compute the probability of finding a word W given a history H, or P (W | H). Such assumption that the probability of a word depends only on the previous word is called Markov assumption [16]. Markov models are a class of probabilistic models that assume that we can calculate the probability of any future unity using only the near past. We can generalize the Bigram (which takes into account only one word history) for Trigram (which takes into account two word history) and thus for the N-gram (which takes into account N - 1 words of the history).

As an example of an N-gram model, suppose that a history H is the sentence "Please turn off your mobile" and we want to compute the probability that the next word is *phone*:

P*(phone | Please turn off your mobile).*

The number of parameters needed to calculate this probability increases exponentially with the number of words in the history H. For the above example we have the following N-gram models:

- *Unigram*: P*(phone)*
- *Bigram*: P*(phone | mobile)*
- *Trigram*: P*(phone | your mobile)*

The algorithm of Pasca and Dienes [17] uses as input corpus a vast collection of Web documents retrieved from Google search engine. First, all n-grams of a specific type for each sentence are computed. Next, a set of anchors (distribution) of each sentence is created and the tuple (Anchors, phrase) is then stored in a list. A count of how many anchors are shared for each pair of

sentences is also done. As in the case of distributional similarity, the greater the number of anchor phrases that are common to two candidates, the greater the likelihood that they paraphrases each other. Finally, the resulting list of paraphrases undergoes a filter in order to eliminate those that are less likely to be similar.

Lin and Pantel [4] discuss how to measure the distributional similarity through paths in dependency trees to induce generalized paraphrases templates as:

$$X \textit{ found an answer to } Y \Leftrightarrow X \textit{ solves } Y$$

A dependency relationship is an asymmetric binary relation between a word called "head" and another word called "modifier". The structure of a sentence can be represented by a set of dependency relationships in a tree structure. A word in this phrase may have various modifiers but, often, each word is the modifier of just another word. The root of the dependency tree does not modify any other word. Figure 2 shows the dependency tree generated for the sentence "Mary found an answer to the problem". Arrows represent a dependency relationship between the head and the modifier whereas the name above them indicates the type of relationship found [4].

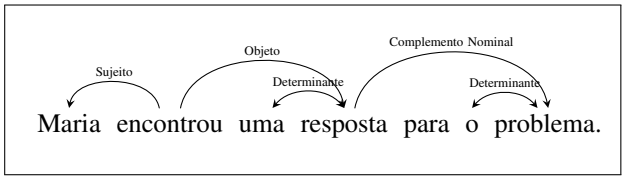

Figure 2. Example of a Dependency Tree

In a dependency tree there are two types of semantic relationship: (i) the direct and (ii) indirect. A direct relationship is represented by the arrows in the figure above (links) and the indirect relationship is represented by a path that connects two words. This path is defined by concatenating dependency relationships with the words found throughout these, excluding the ends. For the sentence of figure 2, the path between words "Mary" and problem" is represented as follows:

N:sujeito:V$\leftarrow$ encontrou $\rightarrow$ V:objeto:N $\rightarrow$ resposta $\rightarrow$ N:para:N,

which means "X found an answer to Y". The words at the extremities of the phrase are called *slots* that are filled with *path context*. For example, the slots X and Y of the template extracted from the path above could be filled with the words "Mary" and "problem", respectively.

Hypothetically, if two paths connecting the same set of words they tend to be similar. Lin and Pantel [4] extend the distributional hypothesis: if two paths tend to occur in similar contexts, thus their meanings tend to be similar.

In order to compute the similarity of the extended distributional hypothesis, it is necessary to compute the frequencies of all paths within the *corpus* and the words that fill their slots. For each instance of a path P which connects two words $W1$ and $W2$, the frequency counter is incremented for both triples $(P, SlotX, W1)$ and $(P, SlotY, W2)$, where $(SlotX, W1)$ and $(SlotY, W2)$ are called *characteristics* of P. The more characteristics two paths divide, more similar they are. It uses a hash table to accumulate the frequency of all features for the whole set of paths extracted from the *corpus*. Essentially, two paths are similar if there is a large number of in common features. [4].

The relationship of mutual information between a path slot and a word that fills it can be computed by:

$$mi(p, Slot, w) = \log\left(\frac{|p, Slot, w| \times |*, Slot, *|}{|p, Slot, *| \times |*, Slot, w|}\right) \quad (1)$$

The similarity between a pair of slots (s_1, s_2) is defined as:

$$sim(s_1, s_2) = \frac{\sum_{w \in T(p_1,s) \cap_T (p_2,s)} X + Y}{\sum_{w \in T(p_1,s)} X + \sum_{w \in T(p_2,s)} Y} \quad (2)$$

where $X = mi(p_1, s, w)$ and $Y = mi(p_2, s, w)$. The similarity between a pair of paths is defined as the geometric mean of the similarities of the slots, such as:

$$S(p_1, p_2) = \sqrt{sim(SlotX_1, SlotX_2) \times sim(SlotY_1, SlotY_2)} \quad (3)$$

The performance of such algorithm is strongly tied to the type of the word that is associated with the root of the extracted path. For example, whereas the verbs tend to have various modifiers, nouns usually do not have more than one. However, if a word has less than two modifiers, it cannot be root for a path. As a consequence, this algorithm tends to perform better for paths whose regards verbs.

III. The RotaFacil system

RotaFacil is an extension of a location mashup system called *RotaCerta* [3]. The main goal is to automate the updating mechanism of POIs database and improve linguistic variability of templates. It is proposed the use of a collaborative basis of POIs and a corpus-driven approach for template generation to achieve such goals. These tasks are described in the following sections along with a detailed overview of system's architecture.

A. System's architecture

The system architecture consists of three modules: (1) Route Generator (GRotas), (2) Reference Generator (GRef) and (3) Text Generator (GText).

The GRotas aims to find a path that connects the origin and destination points and select points along the path so that a pre established minimum distance *md* is preserved. This module uses Google Directions API[4].

The GRef has the following functions: (1) obtain reference points for each route point found by GRotas, (2) find addresses for these points, (3) associate the points to

[4]http://code.google.com/apis/maps/documentation/directions/

the route points, (4) eliminate the useless references and (5) generate an XML file containing all this information.

The GText is responsible for building linguistic textual description for the suggested route. This module consists of two components, namely: (1) the *natural text generator* and (2) the *templates dataset*.

Communication among these modules is done by means of XML files that store information of the route and reference points.

Next, we describe the extension of the modules GRef and GText. In the first, RotaCerta's original POI base has been replaced by the collaborative base *Google Places*. In the second, an approach based on the paraphrases generation have been applied in order to improve the linguistic variability of the generated routes.

B. Automatic feeding of POI database

The GRef is responsible for reviewing the generated route by GRotas and associate to each route point a set of POIs extracted from Google Places.

However, using a collaborative approach leads us to an important issue: how to select the appropriate POI? We have adopted a set of constraints in order to determine if such a POI is useful or not to the route description. The first constraint consists of selecting the references according to a distance X to the point in question. This procedure eliminates remote routes that do not provide an information gain for the textual description. In our experiments, we have empirically defined $X = 50(meters)$. Another important constraint is that only commercial or residencial establishments are valid POIs; neighborhood streets, avenues and parks are not considered. Finally, references that are not at the same address of both points to which they have been attached are also desregarded.

C. Enhancing linguistic variability of templates

A Web application has been developed so to provide a pre-defined set of six different visual routes of Aracaju/SE city, Brazil. Volunteers were then asked to provide a textual description for each route, as if they were guiding someone else. These descriptions constitute the *corpus* of routes used in the creation of the templates. The set of POIs provided by the GRef module was available, so the volunteer could use it to enrich the textual description.

61 route descriptions have been obtained. 36 descriptions have been selected to compose the generation set. The remaining 25 descriptions have been used to evaluate the quality of route descriptions generated by the system. We have defined two different approaches to create the system's set of templates: (1) manual definition and (2) automated generation of templates.

It is worth point out that RotaFacil works with templates and routes in Brazilian Portuguese language. Thus, examples of route descriptions and templates presented in this paper is in Portuguese language. In some sections we provide the English version of the example in order to favor reading and understanding facilities.

Manual extraction of templates

The first method used to identify templates for route descriptions is to manually identify patterns of textual descriptions in the corpus. A pre-processing stage is performed in order to solve eventual grammatical issues. Next, we perform the marking process of the texts, highlighting core data such as street names, avenues and reference points. The purpose of marking is to facilitate recognition of description excerpts that could be used as a template. Figure 3 illustrates such manual process of extracting templates from a route description in Portuguese language. The name of avenues and streets, the points of interests and templates are highlighted.

O senhor vai seguir até o fim dessa rua, que é a *R. Frei Paulo*, chegando na *R. Rafael de Aguiar*, você irá pegar a esquerda e seguir até a **D&E Locação de Veículos**, entrando à esquerda na *Av. Dr. Edésio Vieira de Melo*. Você passará pela **AC Assessoria Contábil**, depois dela são mais 8 ruas até chegar à *Av. Hermes Fontes*. Cruza a avenida, passando pela **Eletrônica Xique**, e irá entrar na primeira rua à direta, que é a *R. Zaqueu Brandão*. Em seguida, entra na primeira rua à esquerda, segue pela *Av. Augusto Maynard*, e quando você passar pelo **O Blazer**, entra na próxima rua à direita. Entra na *Av. Gonçalo Rolemberg Leite*, passa pelo **Centro de Ortodontia Dr. Paulo Soares** e quando chegar no **Bobs Aracaju**, vira à esquerda na *Av. Anízio Azevedo*, passa pela **Fisioline Serv Fisioterapêuticos** e vai entrar na rua da **Feeling**, que é a *R. Homero Oliveira*. A casa dela vai estar na próxima rua após a **Seccaf**, que é a *R. Cristóvão de Barros*.

Label:

Street and Avenues Names
Points of Interest
Templates

Figure 3. Manual extraction of templates uses highlight marks.

It is possible to note that names of streets, avenues and reference points have been replaced by the two specific slots **[STREET]** and **[POI]**. These slots are essential to NLG component of the system. Table I shows the transformation of templates identified from the corpus and those used by the RotaFacil system.

For clarity, Table II shows the English version of the templates for the considered example.

Automatic extraction of templates

For automatic extraction of templates, we have proposed an adaptation of the algorithm of Lin and Pantel [4], for generation of paraphrases. The adapted version of the algorithm is described below.

The algorithm takes as input the dependency trees generated from the corpus of route descriptions.

TABLE I.
THE RESULT FOR THE MANUAL GENERATION OF TEMPLATES
(EXAMPLES IN BRAZILIAN PORTUGUESE LANGUAGE)

Potential Template	New Template
Pegar esquerda e seguir até a **D&E Locação de Veículos**	Pegue à esquerda e siga até o **[POI]**
Quando você passar pelo **O Blazer**, entra na próxima rua à direita	Quando você passar pelo **[POI]**, entre na próxima **[STREET]** à direita
Quando chegar no **Bobs Aracaju**, vira à esquerda na **Av. Anízio Azevedo**	Quando chegar no **[POI]**, vire à esquerda na **[STREET]**
A casa dela vai estar na próxima rua após a **Seccaf**, que é a **R. Cristóvão de Barros**	O seu destino estará na próxima rua após **[POI]**, que é a **[STREET]**
Entrar na rua da **Feeling**, que é a **R. Homero Oliveira**	Entre na rua da **[POI]**, que é a **[STREET]**

TABLE II.
THE RESULT FOR THE MANUAL GENERATION OF TEMPLATES
(EXAMPLES IN ENGLISH)

Potential Template	New Template
Pick up left and follows the **D & E Car Rental**	Pick up left and follows the **[POI]**
When you go through **O Blazer**, enter the next street on the right	When you go through **[POI]**, enter the next **[STREET]** on the right
When you arrive at **Bob's Aracaju**, turn left on **Anízio Azevedo Avenue**	When you arrive at **[POI]**, turn left on **[STREET]**
Her house will be on the next street after **Seccaf**, which is **Cristóvão de Barros Street**	Her house will be on the next street after **[POI]**, which is **[STREET]**
Enter the street where it is located **Feeling Shop**, which is **Romero Oliveira Street**	Enter the street where it is located **[POI]**, which is **[STREET]**

Algorithm 1: Lin and Pantel Algorithm Adapted

Data: List of dependency trees for the corpus (ArvDep)
Result: List of similarities between the paths found in each dependency tree
foreach $tree \in ArvDep$ **do**
  $Paths \leftarrow$ FindPath($tree$)
  Remove of $Paths$ all path with less than 3 words
end foreach
foreach $p_i \in Paths$ **do**
  $Sim \leftarrow$ SimilarityCalculus(p_i, p_{i+1})
  if $Sim > 70\%$ **then**
    $Similarity_{i,i+1} \leftarrow Sim$
    Adiciona $Similarity_{i,i+1}$ na $SimilarityList$
  end if
end foreach
Return $SimilarityList$

Given the dependency trees as input, all the paths (indirect semantic relationships) with less than three words are removed. Next, we calculate the similarity between all remaining paths by means of Lin and Pantel methodology. Finally, for best results, just the paths with similarity values greater than 70% are considered. For results with more than three words *SlotX* has been removed.

In order to illustrate the application of the algorithm, let's consider the description of the route presented in Figure 4. The first step is to extract the dependency tree of this description. For this, we use the tool proposed in [18].

Vire a primeira à esquerda, siga até a Avenida Dr. Edésio Vieira de Melo. Lá, você deve virar para a esquerda novamente. Siga reto e pegue a primeira à direita depois da Av. Hermes Fontes. Logo em seguida, pegue a primeira à esquerda. Quando chegar na Av. Gonçalo Rolemberg Leite, vire à direita e siga reto até a Av. Anízio de Azevedo. Nessa, vire à esquerda e siga até a Rua Homero de Oliveira, que é primeira depois da Av. Acrisio Cruz. Ao final do segundo quarteirão você chegará ao seu destino.

Figure 4. Route description from *corpus*

The dependency tree is thus provided as input to the algorithm described above. Finally, a set of templates is automatically generated as shown in Figure 5. Again, we present the templates in both Brazilian Portuguese and English for clarity.

Siga até o → Siga até o [POI]
Fallow up → Fallow up [POI]

Vire depois de → Vire depois do [POI]
Turn after → Turn after [POI]

Siga até passando → Siga até [STREET] passando [POI]
Follow up through → Follow up [STREET] through [POI]

Figure 5. Automatically extracted set of templates, using the adapted algorithm.

IV. EXPERIMENTS AND RESULTS

Experiments have been divided into two parts. The first part shows how the automatic POI base updating mechanism of RotaFacil solves the huge limitation of RotaCerta's, thus enabling the utilization of POIs for whatever intended Brazilian city in the generation of route descriptions. Different paths for three different cities in Brazil have been generated: Aracaju/SE, Belo Horizonte/MG and Porto Alegre/RS. The second part of experiments aims to evaluate the quality of the text generated by RotaFacil. At this stage we use the similarity of texts based on TF/IDF to compare the route descriptions generated by RotaFacil to the set of route descriptions provided by volunteers and stored as test set. The results of these experiments are presented below.

A. The POI collaborative dataset

City 1: Aracaju/SE

Figure 6 illustrates a route generated in the city of Aracaju/SE. The route includes the ordered streets **Lagarto** and **Lorival Chagas**. The landmarks between the points A and B represent the POIs selected for this route.

Figure 6. Screenshot of RotaFacil system for the route in Aracaju/SE

Figures 7 and 8 show the text generated by Google Maps and the RotaFacil, respectively, for the route of Figure 6. Names in italics represent the descriptions of streets and avenues, and the names in bold represent the POIs selected by the system for the route in question.

Google Maps route description
(Aracaju/SE)

1. Siga na direção sul na *R. Lagarto* em direção à *Av. Barão de Maruim*
2. Vire à direita na *Av. Pedro Paes Azevedo*
3. Continue para *Av. Mariquinha Seixas Dorea*
4. Faça um retorno
5. Pegue a primeira à direita em *R. Lourival Chagas*

Figure 7. Google Maps route description for Aracaju/SE

Note that the textual description provided by RotaFacil (Figure 8) provides a significant amount of references present in the path that allow the user to evaluate the correctness of his/her displacement at every moment.

For example, if we analyze the description number 13 of RotaFacil: "Arrive at destination that is close to **Marlange Hairdressers** in *Lourival Chagas Street*" (translated into English), it is more informative than the corresponding description in google maps (description number 5: Take the first right in *Lorival Chagas Street*").

RotaFacil route description
(Aracaju/SE)

1. Siga na direção sul na *R. Lagarto* em direção à *Av. Barão de Maruim*
2. Passe por **Cardio Imagem** na *R. Lagarto*
3. Passe por **Odonto Consultórios Odontológicos** na *R. Campos*
4. Passe por **INTERDATA SOLUES EM AUTOMAO** na *Av. Augusta Maynard*
5. Vire na *R. Const. João Alves* próximo a **AS Cosméticos**
6. Vire à direita na *Av. Pedro Paes Azevedo*
7. Continue pela *Av. Pedro Paes Azevedo* passando por **UPSIDE COMUNICAÇÃO E GRÁFICA LTDA**
8. Continue pela *Av. Pedro Paes Azevedo* passando por **Pronto Socorro Espiritual Bezerra de Menezes**
9. Siga pela *Av. Mariquinha Seixas Dorea*
10. Continue pela *Av. Mariquinha Seixas Dorea* passando por **EV Projetos e Consultoria LTDA**
11. Continue pela *Av. Mariquinha Seixas Dorea* passando por **SELMA ATELIÊ DE COSTURA**
12. Passe por**Marlange Cabeleireiros** na *R. Lourival Chagas*
13. Chegue no destino que fica próximo à **Marlange Cabeleireiros** na *R. Lourival Chagas*

Figure 8. RotaFacil route description for Aracaju/SE

City 2: Belo Horizonte

Figure 9 illustrates a route in the city of Belo Horizonte/MG. The chosen path is located between Falcatas Street and Federal University of Minas Gerais, at Antonio Carlos Avenue.

Figure 9. Screenshot of RotaFacil system for the route in Belo Horizonte/MG

Again, we present both textual descriptions. In Figure 10, the text generated by Google Maps and Figure 11,

the text generated by RotaFacil with POI information highlighted in bold. Both textual descriptions are for the route shown in Figure 9.

Google Maps Route Description
(Belo Horizonte/MG)

1. Siga na direção leste na *Alameda das Falcatas* em direção à *Alameda dos Coqueiros*
2. Vire à direita na *Alameda das Latânias*
3. Pegue a primeira à esquerda para pegar a *Avenida Coronel José Dias Bicalho*
4. Pegue a primeira à direita em *Avenida Presidente Antônio Carlos*

Figure 10. Google Maps route description for Belo Horizonte/MG

RotaFacil Route Description
(Belo Horizonte/MG)

1. Siga em direção do **New Commerce TI Ltda** na *Alameda das Falcatas*
2. Vire à direita na *Alameda das Latânias*
3. Passe por **Pierrot Fantasies** na *Avenida Coronel José Dias Bicalho*
4. Continue pela *Avenida Coronel José Dias Bicalho* passando por **Comercial Pandoro Ltda**
5. Passe por **Colégio Brasileiro de Medicina Estática** na *Rua Leopoldino dos Passos*
6. Chegue no destino que fica próximo a **CPEJr - Consultoria e Projetos Elétricos Júnior** na *Avenida Presidente Antônio Carlos*

Figure 11. RotaFacil route description for Belo Horizonte/MG

This example clearly shows how the RotaFacil can assist in reading the description of the route. If we look at the description number 1, the text provided by Google Maps (in english: "Go east on Alameda das Falcatas Street toward the Alameda dos Conqueiros Street"), we notice the allusion to cardinal points (*east*) in order to guide the user. Unfortunately, identifying the east in an urban environment, it is not always a simple task. The RotaFacil provide, for the same description, information from reference point in order to facilitate the user's task. Translating the description RotaFacil 1 for English, we have: "Head towards the **New Commerce IT Ltd** in **Alameda das Falcatas Street**", where "New Commerce IT Ltd" is a known commercial establishment, used as POI.

City 3: Porto Alegre/RS

Figure 12 illustrates a route in the city of Porto Alegre/RS. The chosen path is located between two important known city locations: the Estadio Olimpico de Futebol and the Estadio de Futebol Beira Rio.

Figure 12. Screenshot of RotaFacil system for the route in Porto Alegre/RS

The text generated by the query to Google Maps is shown in Figure 13 and the text generated by RotaFacil is described in Figure 14:

Google Maps Route Description
(Porto Alegre/RS)

1. Siga na direção noroeste na *Avenida Coronel Gastão Haslocher Mazeron* em direção à *Rua Catão Coelho*
2. Vire à esquerda na *Rua José de Alencar*
3. Vire à direita na *Avenida Borges de Medeiros*
4. Faça um retorno na *Avenida Praia de Belas*
5. Continue para *Avenida Padre Cacique*. O destino estará à direita.

Figure 13. Google Maps route description for Porto Alegre/RS

The chosen example concerns a route with much more guiding details.

Google Maps description only considers the names of the avenues. However, the avenues have several points of reference that help the user to verify he/she following the right path. For example, the José de Alencar Street referenced by Google Maps in the description of number 2 (in english: "Turn left at *José de Alencar Street*") is described in RotaFacil with a set of reference points such as the description of number 13 (in english: "Turn on *José de Alencar Street* which is near of **SIDI medical clinic**").

B. Assessing the quality of generated text

In this section, we present the results of the evaluation of the quality of texts generated by RotaFacil using both template generation approaches: (i) automatic extraction (RotaFacil AT) and (ii) manual creation (RotaFacil MA). By hypothesis, it is assumed that the RotaFacil MA produces texts that are more similar to the corpus if compared to RotaFacil AT system which is, in turn, assumed to have greater similarity to Google Maps. In other words, the hypothesis is that the generation of routes in natural language using manually built templates is the closest to the way people usually guide other people to move between different points of a city.

RotaFacil Route Description
(Porto Alegre/RS)

1. Siga em direção do **Mini Mercado Sto. Antonio** na *Avenida Dr. Carlos Barbosa*
2. Continue pela *Avenida Dr. Carlos Barbosa* passando por **Mini Mercado Sto. Antonio**
3. Continue pela *Avenida Dr. Carlos Barbosa* passando por **Massolin de Fiori Societa Taliana**
4. Continue pela *Avenida Dr. Carlos Barbosa* passando por **FGTAS-Fundação Gaúcha do Trabalho e Ação Social**
5. Siga pela *Avenida Cascatinha - Medianeira*
6. Passe por **Alemão Lanches** na *Avenida Cascatinha*
7. Vire à esquerda na *Rua José de Alencar*
8. Siga pela *Rua José de Alencar*
9. Continue pela *Rua José de Alencar* passando por **José Ernesto Azzolin Pasquotto**
10. Siga pela *Rua Gonçalves Dias - Menino Deus*
11. Passe por **Berçário e Escola de Educação Infantil Beija-flor** na *Rua Dr. Oscar Bittencourt*
12. Passe por **Farol** na *Rua Grão Pará - Menino Deus*
13. Vire na *Rua José de Alencar* próximo a **SIDI-Serviço de Investigação Diagnóstica**
14. Vire à direita na *Avenida Borges de Medeiros*
15. Siga pela *Avenida Borges de Medeiros*
16. Siga pela *Viaduto Pedro - Praia de Belas*
17. Siga pela *Avenida Borges de Medeiros*
18. Siga pela *Avenida Padre Cacique*
19. Continue pela *Avenida Padre Cacique* passando por **Armazém do Sabor**
20. Chegue no destino que fica próximo a **Zé Pneus** na *Avenida Padre Cacique*

Figure 14. RotaFacil route description for Porto Alegre/RS

The quality measurement of the RotaFacil system has been made by comparing the generated descriptions to the texts of the test set. This is done by computing the similarity between texts. For that, we have used the statistical measure Term Frequency - Inverse Document Fequency (TF-IDF). TF-IDF is a value that represents how a word is relevant for such a document in regards to a collection (corpus). This importance increases proportionally with the number of times the word appears within the document and decreases according to the frequency of the word throughout the collection [19].

TF (term frequency) corresponds to the number of times the term occurs in the document normalized according to the document size and is calculated by:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{4}$$

where $n_{i,j}$ is the number of occurrences of the term *i* in the document *j* and the denominator is the number of occurrences of all terms in the document *j*.

The IDF (Inverse Document Frequency) evaluates the importance of the term in the collection and is given by:

$$idf_i = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|} \tag{5}$$

where $|D|$ is the total number of documents in the corpus and $|\{d_j : t_i \in d_j\}|$ is the number of documents where the word t_i appears. The TF-IDF is calculated by multiplying the equations 4 and 5, such that:

$$tfidf_{i,j} = tf_{i,j} * idf_i \tag{6}$$

Each term within a document is associated to a weight given by the value *TF-IDF*, computed for the whole corpus. Then, each document (d) is represented by a vector ($\overrightarrow{v}$) of real numbers concerning the weights *TF-IDF* of its terms. Given two documents d_1 and d_2, the similarity between them is given by:

$$simCos(d_1, d_2) = \frac{\overrightarrow{v}(d_1).\overrightarrow{v}(d_2)}{|\overrightarrow{v}(d_1)||\overrightarrow{v}(d_2)|} \tag{7}$$

where $\overrightarrow{v}(d_1)$ e $\overrightarrow{v}(d_2)$ are the term vectors of documents d_1 and d_2, respectively.

We have provided 6 route descriptions using the points of origin and destination described in Table III. These routes were the same used for the generation of the corpus. Since each route has more than one description within the corpus, the texts of RotaFacil have been compared with each of these descriptions.

TABLE III.
TYPES OF MAP DESCRIPTIONS

Route	Source Address	Destination Address
A	Rua Geru	Rua Aquidabã
B	Rua Ananias de Azevedo	Rua Ns. Sra. das Dores
C	Rua Frei Paulo	Rua Cristóvão de Barros
D	Rua Lúcio Mota	Rua Moacir Wandeley
E	Rua Manoel Eculides de Oliveira	Rua João Vitor de Matos

The test set consists of 6 descriptions of type A, 5 descriptions of type B, 6 descriptions of type C, 6 descriptions of type D and 2 descriptions of type E, totaling 25 route descriptions. Obviously, route descriptions have been compared within its particular type (same source and destination addresses).

Tables IV, V, and VI show the results of similarity to Google Maps, RotaFacil MA and RotaFacil AT, respectively, when compared with the descriptions of test set.

TABLE IV.
SIMIALRITY BETWEEN GOOGLE MAPS AND THE TEST SET

Route	1	2	3	4	5	6	Average
A	0,01	0,008	0,03	0,02	0,02	0,02	0,02
B	0,10	0,02	0,05	0,01	0,05	-	0,04
C	0,03	0,08	0,11	0,16	0,12	0,01	0,08
D	0,07	0,16	0,09	0,10	0,03	0,12	0,09
E	0,01	0,03	-	-	-	-	0,02

Table VII shows the comparative result between RotaFacil MA, the RotaFacil AT and Google Maps. As seen earlier, each route type has different amounts of

TABLE V.
SIMIALRITY BETWEEN ROTAFACIL MA AND THE TEST SET

Route	1	2	3	4	5	6	Average
A	0,05	0,13	0,15	0,08	0,03	0,17	0,10
B	0,18	0,08	0,07	0,02	0,20	-	0,11
C	0,08	0,07	0,08	0,10	0,12	0,06	0,08
D	0,09	0,17	0,11	0,14	0,12	0,15	0,13
E	0,02	0,05	-	-	-	-	0,03

TABLE VI.
SIMIALRITY BETWEEN ROTAFACIL AT AND THE TEST SET

Route	1	2	3	4	5	6	Average
A	0,04	0,07	0,05	0,10	0,01	0,06	0,06
B	0,14	0,03	0,10	0,03	0,20	-	0,10
C	0,03	0,04	0,14	0,13	0,13	0,10	0,10
D	0,12	0,19	0,15	0,16	0,10	0,12	0,14
E	0,08	0,04	-	-	-	-	0,06

descriptions. In order to compare the systems with each other, the weighted average has been calculated according to the number of descriptions of each route.

TABLE VII.
RESULT OF COMPARISON BETWEEN GOOGLE MAPS, ROTAFACIL MA AND ROTAFACIL AT

Route	Google Maps	RotaFacil MA	RotaFacil AT
A	0.021	0.105	0.06
B	0.049	0.114	0.104
C	0.089	0.088	0.101
D	0.097	0.133	0.143
E	0.026	0.038	0.065
Weighted Average	0.062	0.104	0.099

From the comparison of RotaFacil MA and Google Maps, we see that RotaFacil MA had a weighted average of 10.42% whereas Google Maps achieved 6.2%. It means that the route descriptions generated by the RotaFacil MA is on average 68.64% more similar to the routes generated by people in comparison to Google Maps.

From the comparison of RotaFacil AT and Google Maps, we note that RotaFacil AT had similar performance to that of RotaFacil MA: 9.93% compared to 6.2% of Google Maps, which means 60.16% higher.

We have noticed no significant difference between RotaFacil MA and RotaFacil AT, which points out that the RotaFacil AT is the better choice for the generation of templates due to the lack of need to manually create the identify and extratct the templates from corpus.

V. CONCLUSION

In this paper, we presented the mashup location system RotaFacil. RotaFacil provides natural language descriptions of route, considering Points of Interest (POI). We detail its automated mechanism for automatic feeding of the POI database system and two different approaches for the creation of language templates from a corpus of route descriptions provided by volunteers. A first approach is to manually identify common patterns of text. The second approach is an adaptation of an algorithm for generating paraphrases and enables the automatic identification of such patterns, thus lowering human effort.

The POI base updating mechanism has proved to be effective for generating natural route descriptions regardless of region and independent from human intervention. We present examples for three different brazilian cities.

The corpus-driven approach to provide liguistic templates in support of generation of route descriptions has shown good results. The set of templates has been improved both quantitatively and in terms of linguistic variability. The quality of generation provided by RotaFacil was evaluated by measuring the similarity of generated route descriptions with a test set containing textual descriptions of same routes provided by volunteers. Experiments have shown that both manual and automatic approaches to creation of linguistic templates, proposed in this paper, lead to generation of route descriptions much closer to the way people actually orient others and themselves in their way if compared to Google Maps.

Once the results has shown no significant difference between both approach, we conclude that the automatic approach for creating linguistic templates should be prioritzed, since it considerably reduces human effort.

A known limitation is that although both mechanism for POI base updating and automatic generation of template is language independent, current version of RotaFacil cannot provide route descriptions in any language other than Portuguese. We are working to extend its template base to other languages.

A mobile version of RotaFacil which takes GPS data of user's current location is also being considered.

ACKNOWLEDGEMENT

The authors thank the *Programa Especial de Incluso em Iniciao Cientfica* (Piic/Proest/Posgrap/UFS) for granting a scholarship to Bruno Barroso.

REFERENCES

[1] J. Stewart, S. Bauman, M. Escobar, J. Hilden, K. Bihani, and M. W. Newman, "Accessible contextual information for urban orientation," in *Proceedings of the 10th international conference on Ubiquitous computing*, ser. UbiComp '08. New York, NY, USA: ACM, 2008, pp. 332–335. [Online]. Available: http://doi.acm.org/10.1145/1409635.1409679

[2] M. Duckhama, S. Wintera, and M. Robinsonb, "Including landmarks in routing instructions," *Journal of Location Based Services*, vol. 4, no. 1, pp. 28–52, 2010.

[3] A. Guimaraes and H. Macedo, "A mashup system to generate route descriptions based on points of interest," in *Proceedings of the 5th Euro American Conference on Telematics and information Systems*, 2010, pp. 1–8.

[4] D. Lin and P. Pantel, "Dirt @sbt@discovery of inference rules from text," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '01. New York, NY, USA: ACM, 2001, pp. 323–328. [Online]. Available: http://doi.acm.org/10.1145/502512.502559

[5] M. Batty, A. Hudson-Smith, R. Milton, and A. Crooks, "Map mashups, web 2.0 and the gis revolution," *Annals of GIS*, vol. 16, no. 1, pp. 1–13, 2010. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/19475681003700831

[6] Wikipedia, "Wikipedia," http://en.wikipedia.org/wiki/ (acessado em 24 de outrubro de 2012), 2012. [Online]. Available: http://en.wikipedia.org/wiki/

[7] ProgrammableWeb, "ProgrammableWeb," http://www.programmableweb.com/ (acessado em 13 de novembro de 2012), 2012. [Online]. Available: http://www.programmableweb.com/

[8] Google, "Google Maps API," http://code.google.com/apis/maps, 2012. [Online]. Available: http://code.google.com/apis/maps

[9] E. Reiter and R. Dale, *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

[10] J. D. Moore and J. Oberlander, "Natural language generation: An introduction," 2012. [Online]. Available: http://www.inf.ed.ac.uk/teaching/courses/nlg/lectures/2011/NLG2011Lect1.pdf(acessadoem27deoutrubrode2012)

[11] H. T. Macedo, "A Software Architecture for Ubiquitous Web Browsing with Application to Recommendation Systems," Ph.D. dissertation, Centro de Informtica da Universidade Federal de Pernambuco, 2006.

[12] E. Reiter and R. Dale, "Building applied natural language generation systems," *Nat. Lang. Eng.*, vol. 3, no. 1, pp. 57–87, Mar. 1997. [Online]. Available: http://dx.doi.org/10.1017/S1351324997001502

[13] K. Van Deemter, E. Krahmer, and M. Theune, "Real versus template-based natural language generation: A false opposition?" *Comput. Linguist.*, vol. 31, no. 1, pp. 15–24, Mar. 2005. [Online]. Available: http://dx.doi.org/10.1162/0891201053630291

[14] P. W. Culicover, "Mechanical translation and computational linguistics," in *Paraphrase generation and information retrieval from stored text*, 1968, vol. 11, pp. 78–88.

[15] N. Madnani and B. J. Dorr, "Generating phrasal and sentential paraphrases: A survey of data-driven methods," *Comput. Linguist.*, vol. 36, no. 3, pp. 341–387, Sept. 2010. [Online]. Available: http://dx.doi.org/10.1162/coli-a-00002

[16] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*, 2nd ed. Prentice Hall, Feb. 2008. [Online]. Available: http://www.worldcat.org/isbn/013122798X

[17] M. Pasca and P. Dienes, "Aligning needles in a haystack: Paraphrase acquisition across the web," in *Natural Language Processing ? IJCNLP 2005*, ser. Lecture Notes in Computer Science, R. Dale, K.-F. Wong, J. Su, and O. Kwong, Eds. Springer Berlin Heidelberg, 2005, vol. 3651, pp. 119–130. [Online]. Available: http://dx.doi.org/10.1007/11562214-11

[18] E. Bick, "The parsing system "palavras": Automatic grammatical analysis of portuguese in a constraint grammar framework," Ph.D. dissertation, Aarhus University, 2000.

[19] L. C. G. Maia and R. R. Souza, "Medidas de similaridade em documentos eletrnicos," 2008.

Rafael Teles was born in Aracaju/SE, Brazil, in 1990. He graduated in Computer Science at the Federal University of Sergipe in 2012. His current scientific research focuses on natural language generation.

Bruno Barros was born in Belo Horizonte/MG, Brazil, in 1991. He is an undergraduated student of Computer Science course at the Federal University of Sergipe. His current academic interest relies on natural interfaces for mobile devices.

Adolfo Guimaraes was born in Aracaju/SE, Brazil, in 1984. He graduated in Computer Science at the Federal University of Sergipe in 2009. He obtained a Master's degree in Computer Science from the Federal University of Minas Gerais in 2013. From July 2013 until the present time, he works as a temporary professor at the Departament of Information System, Federal University of Sergipe, Brazil. His current scientific research focuses on natural language processing, recommender systems and genetic programming.

Hendrik Macedo was born in Aracaju/SE, Brazil, in 1977. He graduated in Computer Science at the Federal University of Sergipe in 1998. He obtained a Master's degree in Computer Science from the Federal University of Pernambuco in 2001 and a doctorate in Computer Science also from the Federal University of Pernambuco in 2006, having done an internship PhD at the University of Paris VI in 2002. From July 2006 until the present time, serves as an associate professor in the Department of Computer Science, Federal University of Sergipe, Brazil, where he held the position of vice-coordinator of the Graduate Program in Computer Science at this University. His current scientific research primarily focuses on speech natural interfaces.

Call for Papers and Special Issues

Aims and Scope

Journal of Emerging Technologies in Web Intelligence (JETWI, ISSN 1798-0461) is a peer reviewed and indexed international journal, aims at gathering the latest advances of various topics in web intelligence and reporting how organizations can gain competitive advantages by applying the different emergent techniques in the real-world scenarios. Papers and studies which couple the intelligence techniques and theories with specific web technology problems are mainly targeted. Survey and tutorial articles that emphasize the research and application of web intelligence in a particular domain are also welcomed. These areas include, but are not limited to, the following:

- Web 3.0
- Enterprise Mashup
- Ambient Intelligence (AmI)
- Situational Applications
- Emerging Web-based Systems
- Ambient Awareness
- Ambient and Ubiquitous Learning
- Ambient Assisted Living
- Telepresence
- Lifelong Integrated Learning
- Smart Environments
- Web 2.0 and Social intelligence
- Context Aware Ubiquitous Computing
- Intelligent Brokers and Mediators
- Web Mining and Farming
- Wisdom Web
- Web Security
- Web Information Filtering and Access Control Models
- Web Services and Semantic Web
- Human-Web Interaction
- Web Technologies and Protocols
- Web Agents and Agent-based Systems
- Agent Self-organization, Learning, and Adaptation
- Agent-based Knowledge Discovery
- Agent-mediated Markets
- Knowledge Grid and Grid intelligence
- Knowledge Management, Networks, and Communities
- Agent Infrastructure and Architecture
- Agent-mediated Markets
- Cooperative Problem Solving
- Distributed Intelligence and Emergent Behavior
- Information Ecology
- Mediators and Middlewares
- Granular Computing for the Web
- Ontology Engineering
- Personalization Techniques
- Semantic Web
- Web based Support Systems
- Web based Information Retrieval Support Systems
- Web Services, Services Discovery & Composition
- Ubiquitous Imaging and Multimedia
- Wearable, Wireless and Mobile e-interfacing
- E-Applications
- Cloud Computing
- Web-Oriented Architectrues

Special Issue Guidelines

Special issues feature specifically aimed and targeted topics of interest contributed by authors responding to a particular Call for Papers or by invitation, edited by guest editor(s). We encourage you to submit proposals for creating special issues in areas that are of interest to the Journal. Preference will be given to proposals that cover some unique aspect of the technology and ones that include subjects that are timely and useful to the readers of the Journal. A Special Issue is typically made of 10 to 15 papers, with each paper 8 to 12 pages of length.

The following information should be included as part of the proposal:

- Proposed title for the Special Issue
- Description of the topic area to be focused upon and justification
- Review process for the selection and rejection of papers.
- Name, contact, position, affiliation, and biography of the Guest Editor(s)
- List of potential reviewers
- Potential authors to the issue
- Tentative time-table for the call for papers and reviews

If a proposal is accepted, the guest editor will be responsible for:

- Preparing the “Call for Papers” to be included on the Journal’s Web site.
- Distribution of the Call for Papers broadly to various mailing lists and sites.
- Getting submissions, arranging review process, making decisions, and carrying out all correspondence with the authors. Authors should be informed the Instructions for Authors.
- Providing us the completed and approved final versions of the papers formatted in the Journal’s style, together with all authors’ contact information.
- Writing a one- or two-page introductory editorial to be published in the Special Issue.

Special Issue for a Conference/Workshop

A special issue for a Conference/Workshop is usually released in association with the committee members of the Conference/Workshop like general chairs and/or program chairs who are appointed as the Guest Editors of the Special Issue. Special Issue for a Conference/Workshop is typically made of 10 to 15 papers, with each paper 8 to 12 pages of length.

Guest Editors are involved in the following steps in guest-editing a Special Issue based on a Conference/Workshop:

- Selecting a Title for the Special Issue, e.g. “Special Issue: Selected Best Papers of XYZ Conference”.
- Sending us a formal “Letter of Intent” for the Special Issue.
- Creating a “Call for Papers” for the Special Issue, posting it on the conference web site, and publicizing it to the conference attendees. Information about the Journal and Academy Publisher can be included in the Call for Papers.
- Establishing criteria for paper selection/rejections. The papers can be nominated based on multiple criteria, e.g. rank in review process plus the evaluation from the Session Chairs and the feedback from the Conference attendees.
- Selecting and inviting submissions, arranging review process, making decisions, and carrying out all correspondence with the authors. Authors should be informed the Author Instructions. Usually, the Proceedings manuscripts should be expanded and enhanced.
- Providing us the completed and approved final versions of the papers formatted in the Journal’s style, together with all authors’ contact information.
- Writing a one- or two-page introductory editorial to be published in the Special Issue.

More information is available on the web site at http://www.academypublisher.com/jetwi/.

(Contents Continued from Back Cover)