

Automated integration of transport timetable information^{*}

João Westerberg and Jorge Santos

Instituto Superior de Engenharia do Porto, Porto, Portugal
{1150447,ajs}@isep.ipp.pt

Abstract. The ever-growing Web contains a large amount of data. This large amount of data is useful when combined with applications that can refine it and use it to improve its users' lives. However, using the data available is not an easy task since most of the information is not represented in machine-friendly formats.

In this project, a solution using ontology-based integration combined with web robots' extraction automates the process required for updating information regarding schedules of public transports.

Keywords: Information extraction, Information integration, Ontology mapping, Web scraping

1 Introduction

In a time when society views public transportation as a cost-effective and eco-friendly medium of dislocation [6], there are opportunities for solutions that focus on the publication of relevant information to the topic.

Although public transportation's information is available online, it is scattered throughout the Web in an unorganized and unstructured format. There is a necessity for a tool that extracts the information and transforms it into standardized formats.

This work proposes a solution capable of converting data from public transportation timetables into different structured formats. Most transport providers only display their timetable information on their respective websites. There, the information is portrayed in a non-machine friendly manner since the target audience is human users.

During the research the team analysed the distribution of file format used for the timetables from a sample 25 relevant public transportation agencies, operating in the capital (Lisbon) and the northern region of Portugal. This analysis revealed that 52% of the agencies presented PDFs to present their timetables and other 36% provided this information using HTML. The remaining agencies used either a machine readable format such as JSON or a format irrelevant to the study such as flash.

^{*} Supported by OPT - Optimização e Planeamento de Transportes

Agencies can also use different models to present the information, despite the file format used to distribute the timetables. For example the timetables in figure 1 and 2 represent two different models of timetables where the model in figure 1 shows the stops in the first row and the model in figure 2 shows the stops in a column. Although both agencies represent timetables using a tabular model containing rows, columns, and cells, the different orientation of the information requires a different viewpoint from the customers. While this type of knowledge is subconsciously handled by a human, it becomes a concern for an integration process, resulting in additional requirements for the systems' algorithms.

Estádio do Dragão	Campanhã	Heroísmo	24 de Agosto	Bolhão	Trindade
06:04	06:06	06:08	06:10	06:11	06:12
06:19	06:22	06:24	06:25	06:27	06:28
06:34	06:36	06:38	06:40	06:41	06:42
06:50	06:52	06:54	06:56	06:57	06:58

Fig. 1: Timetable by "metro do Porto"

Porto >> Póvoa de Varzim					
Porto	08:00	11:00	16:00	19:00	
Porto (Hospital S. João)					
Monte Burgos	08:10	11:10	16:10	19:10	
Moreira da Maia	08:25	11:25	16:25	19:25	
Soutelo	08:35	11:35	16:35	19:35	
Vila do Conde	08:50	11:50	16:50	19:50	
Póvoa de Varzim	09:00	12:00	17:00	20:00	

Fig. 2: Timetable by "AVMinho"

The nonexistence of APIs or machine convenient formats such as RDF, CSV, JSON, or XML, results in an inefficient process for updating an information distribution system. Moreover, there is a lack of communication channels between the providers and the other distribution system stakeholders. This lack of channels results in a pull mechanism requirement, in which the system periodically verifies if the information available on the providers' website was changed.

2 Related work

The distribution system described in this document falls into the broader category of Extract Transform Load tools. More specifically, on the topic of Information Extraction (IE). IE is a term that has come to be applied to the activity of automatically extracting configurable sorts of information from an input text [5].

In this particular solution, an ontology represents the knowledge from the tabulated formats in the timetables. The information extracted from the providers' websites is parsed and mapped into the ontology Abox. Each output format conceptual model is then represented as a separate ontology. By using ontology mapping, a process whereby two ontologies are semantically related at the conceptual level [10], the knowledge from the Timetable ontology is transformed into the output format, as represented in figure 3.

The survey [11] provides an extensive literature review of existing approaches for ontology-based integration. A different survey [2] focus in ontology mapping, reviewing different tools, and other works within the topic of ontology mapping.

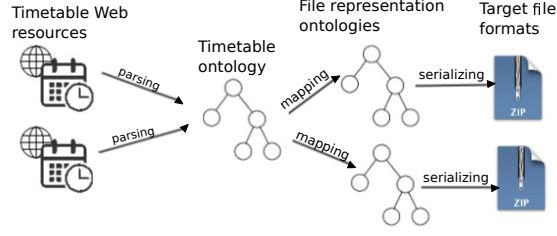


Fig. 3: Mapping pipeline showing the ontologies involved in the format conversion

The techniques used for information extraction are dependent on the type of data source. As previously identified, most timetable information is available at the providers' website as an HTML document or as downloadable PDF files. This creates the need to access those web pages and extract the available documents.

Considering the possibilities that arise from using the Web and its vastness, it should be no surprise that the topic of how to extract user-interested information automatically or semi-automatically has become a research topic from researchers worldwide [12].

The solution uses the Scrapy framework¹ for crawling the providers and scrape data from the timetables, performing HTML extraction. For PDF extraction, an existing PDF extraction project called "Excalibur / Camelot"²[8] was tailored to fit the integration requirements, the domain, and the solution's architecture.

3 Proposed solution overview

The system's operation can be described as a single pipeline, where each execution will integrate the information contained in the websites. The operations in this pipeline follow the logic represented in the activity diagram represented in the figure 4, and are initiated by a scheduled task that verifies if the timetables were updated.

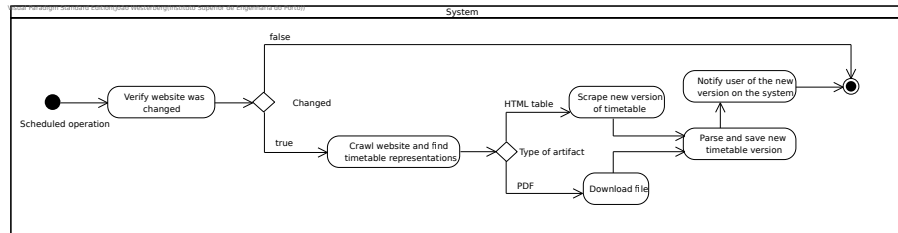


Fig. 4: Activity diagram for high-level view of action in the integration process

¹ Scrapy documentation can be found at: <https://docs.scrapy.org/en/latest/>

² Excalibur can be found at: <https://github.com/camelot-dev/excalibur>

The proposed architecture embodies the following key design principles or design guidelines:

Loose coupling - By creating well-defined interfaces between the different components, it becomes possible to protect each component from external changes. Any changes on one side do not affect the other as long as the interface remained stable.

Language independence - The system uses standard practice in the industry, with a wide array of languages provides implementations such as Web service and messaging. Using these techniques allows for the addition of new components or projects without having to be restricted to the other components language.

Modularity - An advantage of a loose coupling system is that the components can be replaced with an alternative implementation. A well designed modular system not only allows the interchangeability of its parts but also allows for its components to be used in other systems, improving the overall system's reusability.

Reusability - The system ensures reusability through different ways for each component. For instance, the web scraping service provides a set of configurable web scrapers/ spiders. Configuration allows reusing the same spider for different websites of different providers by simply changing a few initial variables. Not only is there an abstraction of the information's source, but there is also an abstraction of the information's domain.

Extensibility - An extensible system allows the addition of new functionalities and assures the reduction of the required time to implement the new functionality. One of the main design choices that improve this system extensibility is polymorphism and dependence injection.

The result is the system represented in figure 5. The communication between the spiders/robots and the Backend happens asynchronously, using the publish/subscribe pattern. In addition, the spider project provides an HTTP API designed for the execution of the spiders.

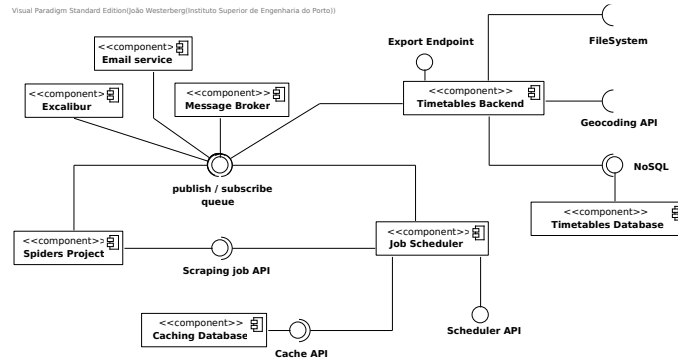


Fig. 5: Coarse grained view of components for proposed solution's system

The system contains a scraping component represented as spidersProject, a component called Job Scheduler that is responsible to periodically schedule scraping requests and provides a facade to the spidersProject API, and a Back-End responsible for the ontology mapping and the output serialization.

The solution can serialize files using the GTFS format³ and a proprietary format. These files can be requested from the Timetables Backend by using a web service represented as "Export Endpoint".

The extracted information is persisted in a NoSQL document database using a format that can be transformed into the ontology's Abox.

When the Timetables Backend receives an export request, the Timetable's instances are retrieved from a database using a repository, and the process of generating individuals is initiated.

3.1 Excalibur design for PDFs' information extraction

The open-source Excalibur project was selected and tailored to fit the solution's requirements for extracting the data and information from the providers' PDFs. Excalibur is a web app, providing a graphic user interface for the camelot library [8], where the camelot library is responsible for performing the extraction algorithm.

The changes made to the forked version of the Excalibur include the addition of asynchronous communication support, a subroutine to download the PDF files, and other changes to the user interface to allow user input related to the timetables service and name fields.

3.2 Asynchronous communication design

The system applies an asynchronous communication approach using messaging with an intermediary component responsible for handling message distribution, also known as a message broker pattern.

This is useful considering that the time of some operations is variable, for example, in a spider execution, it would be unreasonable for a component to actively wait for the recipients' response.

The system uses a RabbitMQ⁴ instance to implement the message broker pattern. Figure 6 represents the exchanges and queues used by the system's components. This is an implementation of the publisher/subscribe pattern, where the publishers are represented in the figure as the producers and the subscribers as the consumers.

As it is noticeable in the diagram from figure 6, the exchanges used in this system are direct exchanges, meaning the communication occurs using multicast via a binding routing key.

³ GTFS is a google maintained data format to represent public transportation timetable and location coordinates

⁴ RabbitMQ documentation is available at: <https://www.rabbitmq.com>

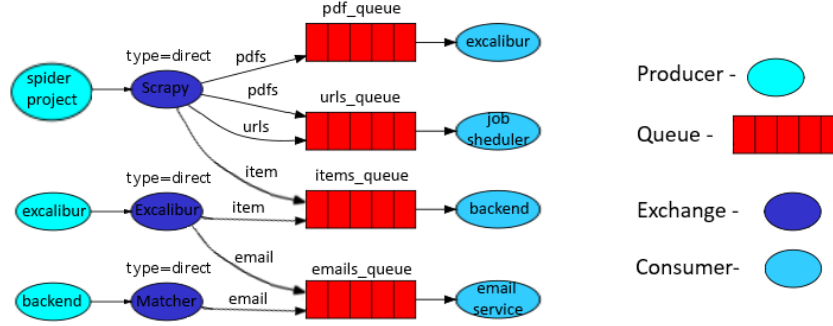


Fig. 6: RabbitMQ exchanges and queues routing

4 Ontology development

The final solution includes three ontologies. Two of these ontologies represent the structure of the desired output files. In contrast, the other ontology works as a bridge connecting the different data formats of the same domain with the file representation ontologies. These ontologies were developed using Stanford's Protege tool [9] and are expressed in RDF/XML [1].

The development of the file representation ontologies was simple. The objective of these ontologies is to represent the classes and taxonomy represented in the desired output. For GTFS, an outdated GTFS ontology⁵ was repurposed and customized to represent the current file formats.

The timetable ontology specifies the format of how this type of data is generally found online, containing axioms and rules for how tabulated data is presented and how a human reader perceives this data under the scope of transport schedules. At the same time, the ontology borrows terminology from the file representation ontologies as necessary.

Using simpler logical axioms from how tabulated data is represented, such as $\text{TableRow} \sqsubseteq \text{hasOrientation}(\text{Horizontal})$, it becomes possible to describe more complex domain-specific axioms and rules such as 1 and 2, implemented using OWL and SWRL respectively. These axioms introduce knowledge to the extract data as it would be perceived in figure 7 if applied to the timetable in figure 1.

$$\begin{aligned} \text{TableComponent} \sqcap (\geq 2 \text{ hasItemInColumn.Stop} \sqcup \geq 2 \text{ hasItemInRow.Stop}) \\ \sqsubseteq \text{StopsTableComponent} \end{aligned} \quad (1)$$

$$\begin{aligned} \forall x, y, z, i : \text{StopTime}(x) \wedge \text{hasTrip}(x, y) \wedge \text{TableRow}(y) \wedge \text{hasColumn}(x, z) \\ \wedge \text{index}(z, i) \rightarrow \text{stopSequence}(x, i) \end{aligned} \quad (2)$$

⁵ GTFS ontology can be found at <http://vocab.gtfs.org/gtfs.ttl#>, credited to Pieter Colpaert and Andrew Byrd

To handle the ontologies, the Backend uses the owlready2 package that integrates the OWL ontology model with the Python object model [7]. The Backend is then able to create the ontologies Abox using an object-oriented approach.

Once the Timetable Ontology Abox contains an agency's instances, defining the mappings between the Timetable Ontology and the OPT ontology and the GTFS ontology is necessary. The set of correspondence/matches between entities of two ontologies are called alignments [3]. A simplified version of the EDOAL language [4] was used to create the alignments. Each alignment is represented by an RDF/XML [1] file containing mappings between the entities of two different ontologies and their relationships.

0	1	2	3	4	5	index
StopsTableComponent						
Estádio do Dragão	Campanhã	Heroísmo	24 de Agosto	Bolhão	Trindade	0
06:04	06:06	06:08	06:10	06:11	06:12	1
06:19	06:22	06:24	06:25	06:27	06:28	2
06:34	06:36	06:38	06:40	06:41	06:42	3
06:50	06:52	06:54	06:56	06:57	06:58	4

Fig. 7: Inferred properties for timetable tabular models

5 Solution evaluation

For evaluating the system, the metrics of Recall and Precision were used by comparing the results of extracting four different agencies with each respective gold standard. The results of these extraction are displayed in table 1 alongside the values for the gold standard and the agencies' basic information. Using these metrics it becomes possible to understand the systems strengths and limitations, and to speculate the quality level of the information retrieved from these sources.

Table 1: Timetable selected for evaluation

Agency	Gondomareense	AVPacense	AVMinho	Rodonorte
Format Type	PDF	PDF	HTML	HTML
True Positives	40	16	58	256
False Negatives	8	0	0	0
False Positives	55	8	0	94
Recall	$\frac{40}{48} \approx 0,833$	$\frac{16}{16} = 1$	$\frac{58}{58} = 1$	$\frac{264}{264} = 1$
Precision	$\frac{40}{95} \approx 0,421$	$\frac{16}{24} \approx 0,667$	$\frac{58}{58} = 1$	$\frac{264}{358} \approx 0.737$

6 Conclusion

This work presented a practical case study where the application of existing information retrieval and integration techniques converges into the engineering, design, and development of a software system capable of converting Web data into information and of integrating the information into an existing system.

The solution proposed combines web scraping and PDF extraction techniques and tools with an ontology based integration, while using a service oriented architecture with asynchronous communication.

The operation of the system resulted in a reduction in the time required to extract the information when compared to a manual approach. Since the process is automated it can also be executed when changes are detected, not requiring additional human resources. At the time, the current solution has a notable difference in extraction quality when comparing the format types. This contrast is expected, since the system extracts every PDF file found in an agency's website, reducing the extraction precision. As such, the most crucial improvement is to provide a process for identifying and categorizing a PDF considering its contents.

References

1. Beckett, D., McBride, B.: Rdf/xml syntax specification (revised). W3C recommendation **10**(2.3) (2004)
2. Choi, N., Song, I.Y., Han, H.: A survey on ontology mapping. *ACM Sigmod Record* **35**(3), 34–41 (2006)
3. David, J., Euzenat, J., Scharffe, F., Trojahn dos Santos, C.: The alignment api 4.0. *Semantic web* **2**(1), 3–10 (2011)
4. Euzenat, J.: Edoal: Expressive and declarative ontology alignment language. URL: <http://alignapi.gforge.inria.fr/edoal.html> (visited on 22/08/2020) (2015)
5. Gaizauskas, R., Wilks, Y.: Information extraction: Beyond document retrieval. *Journal of documentation* **54**(1), 70–105 (1998)
6. Hong, S., Chung, Y., Kim, J., Chun, D.: Analysis on the level of contribution to the national greenhouse gas reduction target in korean transportation sector using leap model. *Renewable and Sustainable Energy Reviews* **60**, 549–559 (2016)
7. Lamy, J.B.: Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine* **80**, 11–28 (2017)
8. Mehta, V.: Camelot documentation release 0.8.2 (2020), <https://readthedocs.org/projects/camelot-py/downloads/pdf/master/>
9. Noy, N., Sintek, M., Decker, S., Crubezy, M., Fergerson, R., Musen, M.: Creating semantic web contents with protege-2000. *Intelligent Systems, IEEE* **16**, 60 – 71 (04 2001). <https://doi.org/10.1109/5254.920601>
10. Silva, N., Rocha, J.: Ontology mapping for interoperability in semantic web. In: *ICWI*. pp. 603–610 (2003)
11. Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-based integration of information-a survey of existing approaches. In: *Ois@ ijcai* (2001)
12. Wei-Guo, Y., Ling-Wei, Y., Ya-Qing, L., Zhi, L.: An ontology-based web information extraction approach. In: *2010 2nd International Conference on Future Computer and Communication*. vol. 1, pp. V1–132. IEEE (2010)