Journal of Engineering Science and Technology Review 14 (6) (2021) 28 - 34

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

Research Article

Dictionary Structure Identification

Lefteris Moussiades[1,*], Ioannis Tsimperidis[1] and Stavros Kanarakis[2]

[1]Department of Computer Science, International Hellenic University, Kavala 65404, Greece
[2]Sword Services Greece SA, Kifisia 14564, Greece

Received 4 June 2021; Accepted 8 December 2021

Abstract

In this paper, we propose a vocabulary structure identification system. The proposed system receives an image of a vocabulary that lies in a textbook, converts the image into text, and then proceeds to identify the vocabulary structure, i.e. it identifies and associates each vocabulary term with its translation, explanation, examples and other relevant information. Our approach is based on pattern matching and the concept of publisher specifications, which represents the vocabulary structure followed by the publisher. Variations in the structure between terms in the same vocabulary are dealt with using directed cyclic graphs. Our experimentation shows positive results and encourages us to further development.

Keywords: Information extraction, OCR image, vocabulary parser

1. Introduction

This paper aims to create electronic vocabularies from lexicographical terms found in English language learning books. Our objective is to use structured vocabularies as “raw material” in other educational applications.
In foreign language textbooks, each section includes a subsection entitled Vocabulary. However, this title can be relatively misleading as this subsection has a dictionary structure. More specifically, it includes the new words that the student should learn. Each word contains its translation into the student's language, its explanation in the foreign language, examples of its use, and other information. Our purpose is to recognize the structure of such a vocabulary to digitize it. The main difficulty in this project is that each publishing house adopts a different vocabulary structure. This is where our contribution in relation to existing work on digitizing dictionaries presented in section 2 focuses.

This work has been implemented for English to Greek vocabularies. However, it is language independent and can be applied to any dictionary involving languages written from left to right.
The steps to digitize a vocabulary are:
1. Initially, a photo is taken from a book page containing the lexical terms.
2. The photo is subjected to pre-processing to improve some of its quality features before it is sent for OCR.
3. The edited photo is then sent to an OCR service to create an RTF file.
4. Finally, the algorithm proposed in this paper is applied to the RTF file to identify the vocabulary structure and its content, i.e., to identify the term for each vocabulary entry, its translation, explanation, examples of usage, etc.

The output of our approach is a JSON file. Early experimental results of our system are encouraging as they show that vocabularies are recognized correctly with high reliability.

The rest of this paper's structure is as follows: Section 2 presents relevant bibliography. A detailed presentation of our approach is given in section 3. Following, in section 4, we offer experimentation, and finally, in section 5, conclusions from this process's implementation are outlined, and proposals are made to develop further and exploit it.

2. Relevant Bibliography

Creating structured text from sources that contain it in amorphous form is achieved through “Information Extraction” technology (IE) [1, 2, 3].
Information Extraction (IE) is a process where structured information is automatically generated from one or more texts that are fully or partially unstructured. Although emerging as a concept in the 70s, it attracted much interest from the scientific community in the late 80s and early 90s with a DARPA-funded project at Message Understanding Conferences (MUC) [1]. IE is applied in a wide range of scientific fields such as biomedicine, finance, information analysis, web search etc. [4]. Sometimes, it's considered the same with Natural Language Processing (NLP) or with Information Retrieval (IR) because it may borrow techniques from the above technologies or it can be used as an intermediate stage by them. However, it must be said that IE deals mainly with the structure of a text, whereas in IR, the text is usually regarded as a Bag of Words (BOW) [5], as well that the goal of IR is to select a subset of texts from a more extensive set. IE systems create structured information, either a summary or a collapsed text derived from the original or the original text in a structured format (database, JSON file, etc.). Beyond that, the result can be introduced into new data mining processes (DM), NLP, or any other function.

In the simplest case, information is extracted by a set of rules applied to the unstructured text. [3]. Most advanced implementations involve using machine learning algorithms so that the “extraction algorithm” can gradually be more efficient and accurate in producing results [2].

*E-mail address: lmous@cs.ihu.gr
ISSN: 1791-2377 © 2021 School of Science, IHU. All rights reserved.
doi:10.25103/jestr.146.03

Depending on the problem's nature, IE systems fall into two categories, single-slot and multi-slot. Single-slot systems extract information related to one object per text. In contrast, multi-slot systems extract information on multiple items [6]. Open IE Systems, a subcategory of IE, create automatically the rules based on the type of information that the system audit with typical examples [7, 8]. IE systems that processing partially structured or unstructured text can be found in [6, 9, 10, 11, 12].

WHISK is a supervised machine learning software that extracts sentence-level information. It needs an input of preconfigured text and semantic tags in words or phrases to train it for the correct identification of text or words in unstructured documents [6]. JASPER [9] uses “template rules” to extract information from limited-range texts. Rapier algorithm [10] consists of an alternative approach to machine learning systems by implementing an inductive logic programming (ILP) system. It is a machine learning system based on a particular set of attributes that extracts information by combining logical rules and background knowledge extracted at the first pass of reading the text. The SRV machine-learning algorithm uses generic features to extract information to operate on a broader range of different text, but it is closer to IR systems than IE. Kushmerick [12] presents a system of “wrapper” classes trained from an inductive learning system.

WOE [7] is an Open IE system that uses an information extractor to extract a set of “triple information” ({*arg1*, *rel*, *arg2*}) from an unstructured text, with *rel* being a rule denoting the semantic relationship between *arg1* and *arg2*. WOE is an unsupervised learning information extractor. Therefore, it can handle several Web resources, unlike TextRunner [8], supervised machine learning that extracts the triple information set only from Wikipedia texts.

IE systems based on machine learning algorithms or Open IE systems require robust computing power to function correctly, making their use on portable systems difficult. Also, the rules from which information are extracted are usually in the form of regular expressions which may be aided by grammar rules or rules that attempt to derive a semantic relationship between words or expressions [6].

Ferreira et al. [13] leveraged MedInX to give structure in narrative data available in electronic form, achieving an F-measure at 0.95. Uddin et al. [14] have presented an algorithm for information and relation extraction to facilitate students' locating important terms of a text and their relationships. In their method, first, they extract the concepts from eBooks, then filtered the critical concepts, subsequently extracted the relations between each couple of concepts, and finally annotated the unstructured text with tags to be more navigable and usable. They applied their method to a set of 30 eBooks and reported excellent results.
Having similar goals, Wang et al. [15] suggested a process for creating a concept hierarchy of a book, which is a powerful tool in presenting and organizing knowledge.

Webpages and their content are also areas in which information extraction is applied. Web wrappers are software that extracts specific information from a webpage. Traditional web wrappers must be adapted to the template of each webpage. Cogar et al. [16] proposed a convolutional neural network to extract structured information from webpages regardless of their template. They use the web rendering engine to obtain screenshots and the DOM tree of a webpage. The neural network then processes the visual and textual content to decide whether a page corresponds to a class belonging to a predefined class set.

In another work, Romero et al. [17] focused on extracting structured information from handwritten texts. Specifically, they used as a dataset 1,747 marriage licenses written in ancient Catalan in the 17th century. In their methodology, n-gram models are just a subclass of Probabilistic Finite-State Machines (PFSM) while using a framework, known as Morphic Generator Grammatical Inference (MGGI), to form constraints. The semantic categories that were defined were the groom's name, the bride's name, the names of their parents, their cities of origin, etc. The results showed a correct extraction of the information with high precision and recall.

Despite the rich literature on extracting structured information from text, none of the above tasks is suitable for our purpose. The works mentioned immediately next are more closely related to our pursuit.

Sassolini et al. [18] attempted to digitize an authoritative historical Italian dictionary consisting of 22,700 pages divided into 21 volumes, containing 183,594 entries. They followed a process for extracting and structuring dictionary contents and converting them into Text Encoding Initiative XML, which has been organized into several iterative steps to reduce the number of unavoidable errors. In their methodology, they first used OCR to recognize the text from the printed form, thus creating word (.doc) format files, then segmented these files so that each segment consisted of 50-60 pages, and continued with the identification of entries. For each identified lexical entry, the headword (or lemma) and a text area corresponding to the body of the entire entry is recognized. The further steps include the iterative segmentation of the body of the lexical entry into different blocks with grammatical information, primary senses, sense attestations and examples, other numbered sub-senses with examples, and etymology. They decided to follow an approach based on pattern matching for the lemma extraction. A check on the number of conditions satisfied is activated whenever the lemma cannot be recognized with certainty. In their work, authors mainly focused on the early stages of converting the dictionary contents into structured digital data. However, as the authors note, the process they propose is not automated but instead requires manual work while simultaneously being incomplete. It is therefore not suitable for our pursuit.

Reul et al. [19] dealt with the typographical variations of the text found in a lexicon and coded the information. For example, changing the font may signal the separation between the lemma and its explanation. For this reason, authors used a German dictionary from the 19th century, which comprises a particularly complex semantic function of typography. The method that followed involved a preliminary phase in which the text would be scanned to convert from several columns to one. Then, a step in which the OCR tool would be modified to recognize 19th-century typographic fonts, a phase in which two models would be trained, one of which would identify the typography and the other the characters, and finally a phase which would combine the results of the two models. Despite the difficulties, such as that the dictionary fonts used are not encountered today, the experimental results showed an error in the correct recognition of the characters of the order of 0.4% and a success rate in recognizing the typography of the order 99%. Although applied to a dictionary, this work is limited to identifying characters that are expressed in fonts that are not currently available. Therefore, it is not suitable for our purpose.

In a similar work Stain [20] turned older printed dictionaries into more easily accessible and more sustainable lexical databases. For this reason, he used a dictionary of the early 20th century with lemmas in Old French and word senses in German, while the only tool available was a list of French lemmas. With the help of OCR, the text was converted to electronic format and manual corrections were made where the double-column format could not be recognized. Then for each of the lemmas available in the list, its location in the text was identified; for each of the lemmas, the senses were extracted, which were identified based on the differences in the typography, such as that they were in italics. Then, for the modernization of the dictionary, Stain linked the extracted senses to a semantic network, specifically the GermanNet. At the same time, for its internationalization, he connected the results with a semantic network in English, WordNet. This work on semantic evaluation is based solely on typographical differences. However, the vocabularies in foreign language textbooks take advantage of typographical differences but do not rely solely on them.

Belyaev et al. [21] deal with almost the same problem, but with a dictionary written in Ossetic, an Iranian language spoken in the Caucasus by approximately 500,000 people. They acknowledged that digitization for dictionaries written in widely spoken languages, such as English, had already taken place. In contrast, many dictionaries remain only in print for other spoken languages, especially minority languages. The authors used off-the-shelf software to help them separate the parts of an entry, such as the headword, one or more senses, one or more examples, etymology, etc. After this, the next stage was converting dictionary format to Text Encoding Initiative (TEI), which defines a set of tags and constraints for representing texts in digital form. The end result was the translation of the dictionary into English and providing semantic markup that would make it searchable across multiple types of data and accessible for machine-based processing. This work concerns a specific lexicon and considers its structure defined and known, so it can not be used to identify dictionaries whose structure varies, such as the vocabularies contained in foreign language books.

Bertrand et al. [22] also deal with the issue of automatic document structuring in their work, recognizing that typography, such as bold or italic letters, plays a crucial role in semantics and can provide important information about the document structure. Their two-phase method was applied to several list-like historical documents, such as dictionaries, catalogues, etc. In the first phase, an entire page is taken as input, and with the help of a convolutional neural network, the words in it are identified. In the second phase, with the help of a deep learning API, each word is classified into one of three classes, i.e. bold, italic, and regular. The experimental results showed that it is possible to identify the typography with an F-score of 0.90. This work is also based solely on typographical differences and is therefore unsuitable for our purpose.

The rest of this paper's structure is as follows: A detailed presentation of our approach and implementation of this process is presented next. Following is the section in which the experimental results of this process are presented. Finally, conclusions from this process's implementation are outlined, and proposals are made to develop further and exploit it.

3. Our Approach

Vocabulary parser (VP) is a component of a larger project that offers an alternative way for kids or adults to learn English. VP allows automatic parsing of contextual information (vocabulary items including terms, explanation, examples etc.) from OCRed documents. Input to VP is an RTF (Rich-Text-Format) document, a ubiquitous output format for documents processed by OCR software.

3.1. Image pre-processing and OCR

The following two subsections present the process of editing the image before it is sent for OCR and the OCR process from which the .rtf file is created, further analyzed.

3.1.1 Pre-processing

Before sending the image for OCR, it must first be edited to improve some of its features to get the best possible result. According to [23, 24], the critical elements that need to be looked at and corrected in an image are contrast, pixel noise, align of characters, color, and file type.

The file type used is .jpg because of its popularity, as it is usually the default type of image storage on most imaging devices. However, other file types provide better quality, e.g. TIFF, PNG [23].

Although it is recommended to convert the image to black and white, or grayscale, to get better OCR results [23, 24], we don't alter it because we use the color of certain words as a feature.

We use the openCV framework to correct pixel noise and contrast [25, 26]. Specifically, among blur, gaussian blur, medianBlur, and bilateral filter, the latter was used to smooth the image noise due to its better results in improving image quality.

Four algorithms [27, 28, 29, 30] were tested regarding the alignment of the text lines. The algorithm described in [28] provided the best results in terms of the alignment process's speed. The original implementation code of the above algorithms is available at [31].

3.1.2 OCR process

We need an OCR capable of working with Java. Out of the available options, we highlighted the cases of ABBYY OCR cloud SDK [32], Tesseract (Tess4J) [33] and Asprise [34]. According to [24], in the paid software category, ABBYY is the best one, while in the free software category, it is Tesseract. We eventually used ABBYY's service, but in the future, we may even try Tesseract's free software, which has excellent reviews, although it has a more complicated setup.

3.2. Publisher Specifications

VP strives to be as generic as possible. Attempts to deal with the fact that the publishers of foreign language educational books use different formats in vocabulary presentation. To accommodate this variation, VP introduces the concept of Publisher Specifications (P-Specs), which encapsulate the exact vocabulary format followed by a specific book publisher. Each publisher has its P-specs created outside of the VP and is fed into the system before processing book pages. For this reason, VP contains an internal Publisher Specs Registry (PSR), which stores all the injected-into-the-system P-specs. For the VP to process an OCR document created from a publisher's book page, the P-specs for that publisher must be in the registry.

Having externalized the format specifications declaration of each publisher, VP offers excellent flexibility and extensibility. By correcting existing specifications, we can enhance the VP's capabilities with zero or minor code modifications. To introduce a new Publisher into the system, we must carefully craft a new P-specs document reflecting the vocabulary format and inject it into the registry. P-specs are currently written in YAML format for the Proof of Concept.

However, this is an implementation detail that could be changed at a later stage. What's important here is that we can add, change or remove P-Specs files to the system. Any new addition to the registry will be loaded automatically, and the subsequent processing cycle can leverage these new specifications.

3.3. Multiple Passes

To make VP more user friendly, processing a document does not require the user to provide information such as publisher name. Instead, VP tries to recognize the publisher from the document itself. The publisher's recognition is being achieved by PSR, which holds all the details about each publisher's vocabulary format. Initially, the RTF document is parsed using a third-party library to extract the information needed for further processing. After that, information takes the form of a stream of strings representing various tokenized vocabulary parts, where each part could be a single token like a word or multiple tokens like a translation phrase or an example sentence.

The stream is then processed twice. In the first pass, the system can recognize the format and mark P-Specs from the registry as active. Once an active publisher has been identified, a second pass consumes the streamed vocabulary parts to recognize valuable vocabulary items. Validity relies on the specific publisher's specifications context.

A more elaborate diagram showing the complete VP high-level architecture and use case flow is shown in Figure 1.

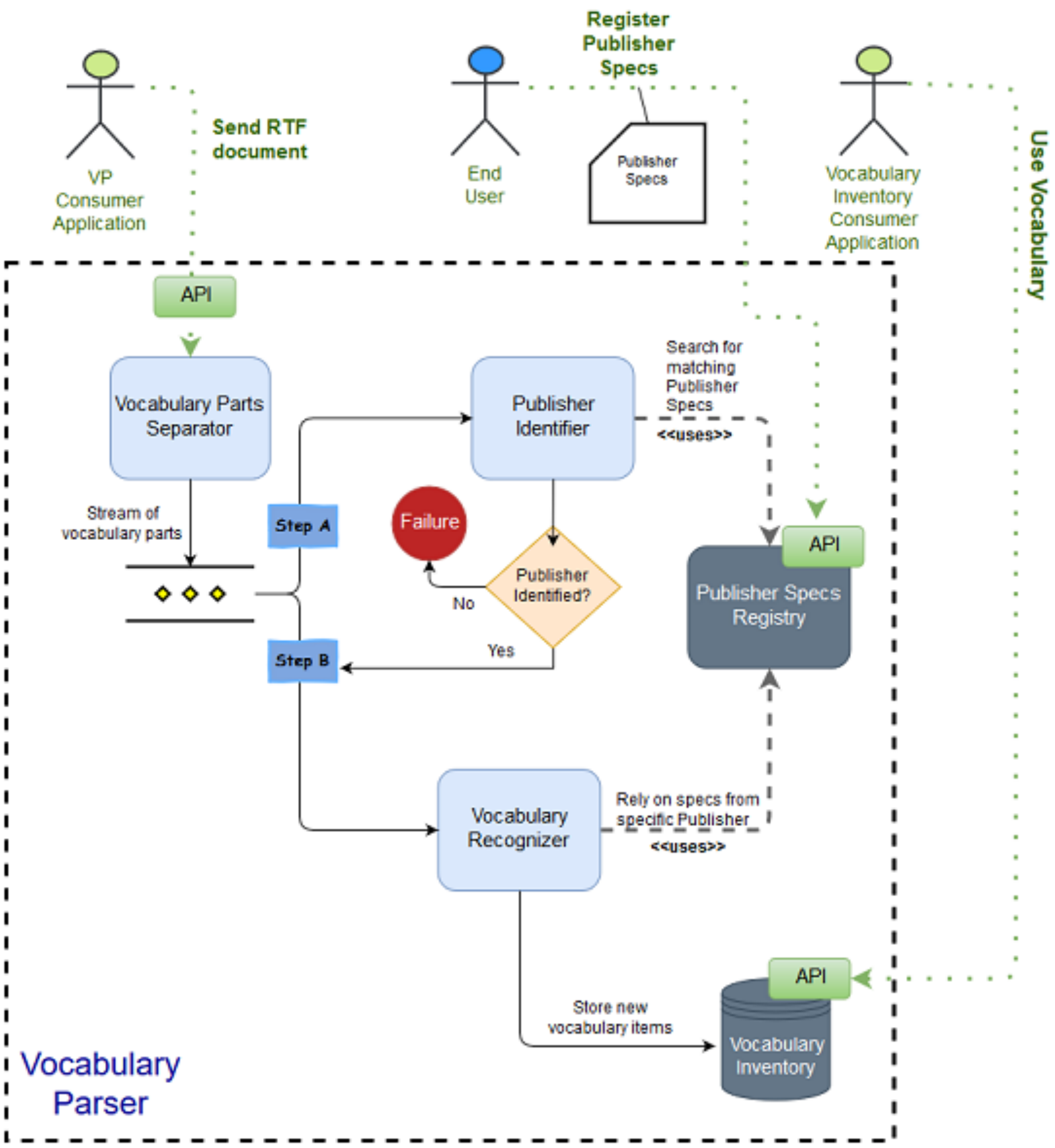


Fig. 1. High-level architecture VP diagram

3.4. Exposed API

As shown in the diagram of Figure 1, VP exposes three different API segments:

- API to pass a new RTF document for parsing, which is used by VP consumer application.
- API to register/unregister/modify Publisher Specifications, used by <project-name> human operator – end user.
- API to consume the inventoried vocabulary items after successfully document processing, which is used by VP vocabulary-consuming applications.

3.5. Vocabulary Format Variations

Vocabulary formats tend to be quite complicated. There may be optional parts, and each term might have a slightly different format between various publishers. We introduce a

processing mechanism that relies on Directed Cyclic Graphs (DCG) to cope with these problems. The construction of a suitable DCG for each publisher comes from the P-Specs context. For example, suppose that a publisher has the following format for its vocabulary items where black color represents mandatory parts whereas grey color represents optional parts
TERM, GRAMMAR_TYPE, TRANSLATION, EXAMPLE, DERIVATIVES

Then, the system will create a DCG, as shown in Figure 2.

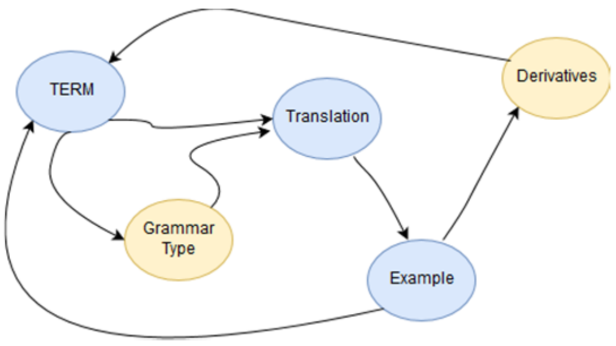
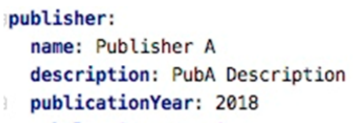


Fig. 2. DCG sample

Processing will always begin from the TERM vertex and move forward in the graph. In cases where there is more than one potential transition to follow, we must resort to extra help for correctly identifying the path. This is accomplished by using regular expressions describing the character pattern that each vocabulary part follows. These regular expressions exist as part of the publisher specifications. Being able to differentiate between different parts for each vocabulary item in a specific publisher is quite challenging. P-Specs editor has to ensure regular expressions are very accurate. Otherwise, the system will fail to choose a correct path transition and thus extract semantically valid vocabulary items. We have been following this approach in the Proof of Concept. Still, if it leads to high levels of complexity or functional errors (in case of undetermined path selections) due to lack of robustness, other methods should be investigated.

3.6. Publisher Specifications

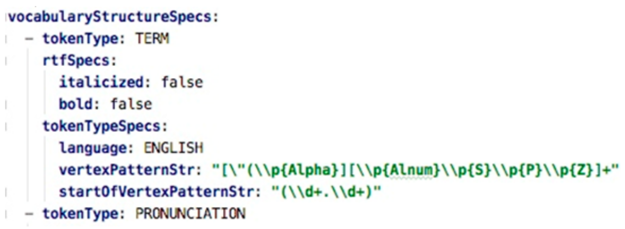
Publisher specifications for the PoC are based on YAML format. In the beginning, it contains publisher metadata, as shown in Figure 3.



```
publisher:
  name: Publisher A
  description: PubA Description
  publicationYear: 2018
```

Fig. 3. Publisher metadata

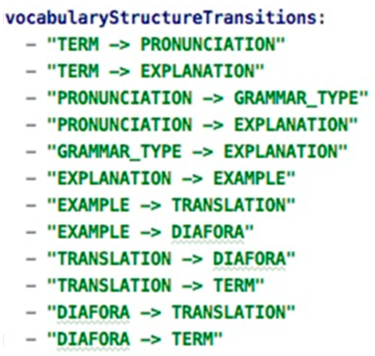
It contains specifications for each vocabulary part item as found in the publisher (Figure 4). In this example, we can see that we declare two sub-types of specifications. The specifications reflecting RTF format-related data for a specific vocabulary part and those reflecting structure-related data (such as language, pattern etc.). We can see that the regular expression pattern that will help us recognize the vocabulary in the second pass of the processing is in the tokenTypeSpecs section of the YAML file for each vocabulary part.



```
vocabularyStructureSpecs:
  - tokenType: TERM
    rtfSpecs:
      italicized: false
      bold: false
    tokenTypeSpecs:
      language: ENGLISH
      vertexPatternStr: "[\"(\\p{Alpha}][\\p{Alnum}\\p{S}\\p{P}\\p{Z}]+"
      startOfVertexPatternStr: "(\\d+.\\d+)"
  - tokenType: PRONUNCIATION
```

Fig. 4. Specifications of each vocabulary part

To construct an appropriate graph representing exactly all the potential vocabulary part transitions, we have the following YAML section, as shown in Figure 5.



```
vocabularyStructureTransitions:
  - "TERM -> PRONUNCIATION"
  - "TERM -> EXPLANATION"
  - "PRONUNCIATION -> GRAMMAR_TYPE"
  - "PRONUNCIATION -> EXPLANATION"
  - "GRAMMAR_TYPE -> EXPLANATION"
  - "EXPLANATION -> EXAMPLE"
  - "EXAMPLE -> TRANSLATION"
  - "EXAMPLE -> DIAFORA"
  - "TRANSLATION -> DIAFORA"
  - "TRANSLATION -> TERM"
  - "DIAFORA -> TRANSLATION"
  - "DIAFORA -> TERM"
```

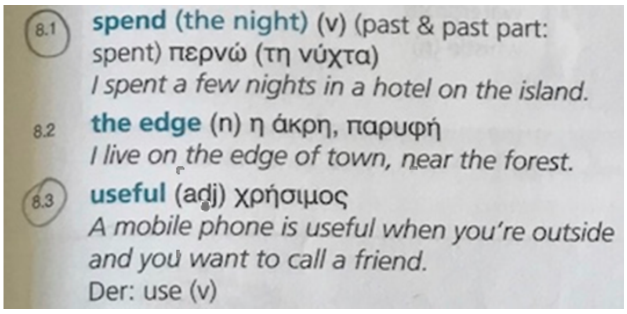
Fig. 5. Structure for the creation of the graph

These transitions will be parsed, and the corresponding graph will be created upon Publisher Specs registration.

4. Experimentation

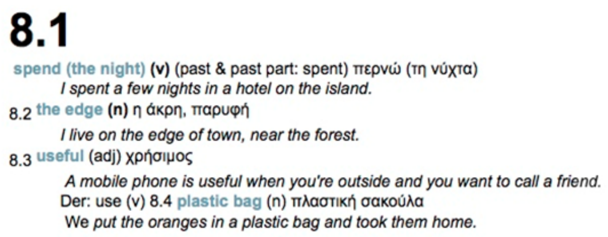
The dataset used to test the procedures described above consists of 14 images taken from two different books. The photos were taken indoors from a mobile phone with automatic settings and under daylight conditions. Some pages had handwritten notes. Also, some of the pages had the characteristic curvature that exists when a book is opened on a table. We mention this because this curvature distorts the layout of the page content. The result is that the text lines are not entirely aligned with each other, which negatively affects accurate text recognition in some cases.
The OCR process is critical because the characters, punctuation marks, and character formatting elements used for IE must be correctly depicted in the RTF file. We initially thought that some formatting elements, such as the size and type of the font and its color, are essential in the analysis process. Still, our tests so far have shown that it is not always possible to obtain this information correctly during the OCR process. For this reason, we currently focus only on features that can be accurately and reliably obtained. When characters, punctuation marks, and formatting features are correctly embedded into the RTF file, the parsing process is considered 100% successful. On the contrary, when there are problems, the process produces incorrect results. So far, the RTFs produced at this stage are not capturing the content 100% correct, so to do the parsing process correctly, we must correct them manually.

In the 14-page sample, 378 different lexicographical terms are 100% successfully recognized and transformed into a structured format (JSON type) in the manually corrected content. Figure 6 shows a sample of the content of one of the sample images, and Figure 7 shows a sample of the content of an RTF file resulting from the OCR procedure, and Figure 8 shows the corresponding content in a structured form.



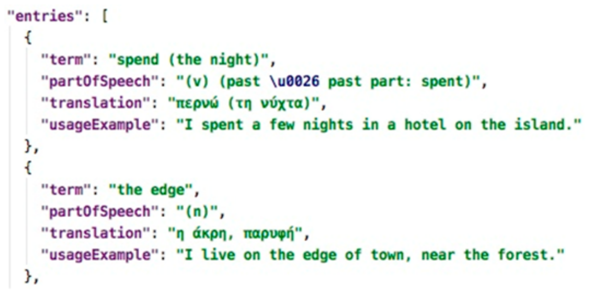
8.1 spend (the night) (v) (past & past part: spent) περνώ (τη νύχτα)
I spent a few nights in a hotel on the island.
8.2 the edge (n) η άκρη, παρυφή
I live on the edge of town, near the forest.
8.3 useful (adj) χρήσιμος
A mobile phone is useful when you're outside and you want to call a friend.
Der: use (v)

Fig. 6. Content sample photo



8.1
spend (the night) **(v)** (past & past part: spent) περνώ (τη νύχτα)
I spent a few nights in a hotel on the island.
8.2 the edge **(n)** η άκρη, παρυφή
I live on the edge of town, near the forest.
8.3 useful (adj) χρήσιμος
A mobile phone is useful when you're outside and you want to call a friend.
Der: use (v) 8.4 plastic bag (n) πλαστική σακούλα
We put the oranges in a plastic bag and took them home.

Fig. 7. Content of RTF file



```
"entries": [
  {
    "term": "spend (the night)",
    "partOfSpeech": "(v) (past \u0026 past part: spent)",
    "translation": "περνώ (τη νύχτα)",
    "usageExample": "I spent a few nights in a hotel on the island."
  },
  {
    "term": "the edge",
    "partOfSpeech": "(n)",
    "translation": "η άκρη, παρυφή",
    "usageExample": "I live on the edge of town, near the forest."
  },
```

Fig. 8. Content of JSON file

There are cases where we can only complete the parsing process with regular expressions. However, this is not always possible, so we use regular expressions in conjunction with formatting text features to implement parsing. In this way, we can do parsing to texts in which the words we have to distinguish have similar characteristics regarding the type of regular expression that they are recognized.

5. Conclusions and Further Development

The results from the experiments conducted in this sample of the documents are very encouraging, demonstrating that the IE in the manner described above is feasible, even though it is still at an early stage.

However, we need to perfect the process of creating RTF files from images as our next step.

Enriching the existing process with functions of finding text that does not meet the specified specifications and isolating it so that the parsing process can continue properly on the remaining text is another step in creating a more flexible and functional parsing process.

Another step is to analyze sample texts from a sufficiently large number of different publishers to disclose the standard features. In this manner, we may create a general specification file used to parse text by many other publishers.

Another direction is creating a machine learning algorithm that will analyze texts to develop the specification file automatically. Creating an algorithm that merges the information contained in different specification files into one is another implementation step towards automating the IE process we have described.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License.

References

1. R. Grishman, "Information extraction: Techniques and challenges", in M.T. Pazienza (eds) "Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology", SCIE 1997, Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 1299, Springer, Berlin, Heidelberg, 11 (1997). doi: 10.1007/3-540-63438-X_2.
2. S. Sarawagi, "Information extraction", Foundations and Trends in Databases **1**(3), 261 (2007). doi: 10.1561/1900000003.
3. S. G. Small and L Medsker, "Review of information extraction technologies and applications", Neural Computing and Applications **25**(3–4), 533 (2014). doi: 10.1007/s00521-013-1516-6.
4. C. C. Aggarwal and C. Zhai (Eds). "Mining text data", Springer, Boston, MA, (2012). doi: 10.1007/978-1-4614-3223-4.
5. J. Cowie and Y. Wilks, Information extraction (1996). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.6480 &rep=rep1&type=pdf.
6. S. Soderland, "Learning information extraction rules for semi-structured and free text", Machine Learning **34**, 233 (1999). doi: 10.1023/A:1007562322031.
7. F. Wu and D. S. Weld, "Open information extraction using Wikipedia", Proc. of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, pp. 118–127 (2010).
8. A. Yates, M. Banko, M. Broadhead, M. Cafarella, O. Etzioni, and S. Soderland, "TextRunner: Open information extraction on the Web", Proc. of Human Language Technologies: The Annual Conf. of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), Rochester, New York, USA, pp. 25-26 (2007).
9. P. M. Andersen, P. J. Hayes, A. K. Huettner, L. M. Schmandt, I. B. Nirenburg, and S. P. Weinstein, "Automatic extraction of facts from press releases to generate news stories", Proc. of the Third Conf. Applied Natural Language Processing, Trento, Italy, pp. 170-177 (1992) . doi: 10.3115/974499.974531.
10. M. E. Cali and R. J. Mooney, "Applying ILP-based techniques to natural language information extraction: An experiment in relational learning", Working Notes of the IJCAI-97 Workshop on Frontiers of Inductive Logic Programming, Nagoya, Japan, pp. 7-11 (1997).
11. D. Freitag, "Toward general-purpose learning for information extraction", Proc. of 36th Annual Meeting of the Association for Computational Linguistics and 17th Int. Conf. Computational Linguistics, Volume 1, Montreal, Quebec, Canada, pp. 404-408 (1998). doi: 10.3115/980845.980914.
12. N. Kushmerick, D. S. Weld, and R. B. Doorenbos, "Wrapper induction for information extraction", Proc. of 15th Int. Joint Conf. Artificial Intelligence, Nagoya, Japan, pp. 729-737 (1997).
13. L. Ferreira, A. J. S. Teixeira, and J. P. Silva Cunha, "Medical information extraction in European Portuguese", in M. M. Cruz-Cunha, I. M. Miranda, and P. Goncalves (eds) "Handbook of Research on ICTs for Human-Centered Healthcare and Social Care Services", Vol. 2, IGI-Global, 607 (2013).
14. A. Uddin, R. Piryani, and V. K. Singh, "Information and relation extraction for semantic annotation of eBook texts", in S. M. Thampi, A. Abraham, S. K. Pal, and J. M. Corchado Rodriguez (eds) "Recent Advances in Intelligent Informatics: Proceedings of the Second International Symposium on Intelligent Informatics", Springer, Cham, Switzerland, 215 (2014).
15. S. Wang, C. Liang, Z. Wu, K. Williams, B. Pursel, B. Brautigam, S. Saul, H. Williams, K. Bowen, and C. L. Giles, "Concept hierarchy

extraction from textbooks", in C. Vanoirbeek and P. Geneves (eds) "Proceedings of the 2015 ACM Symposium on Document Engineering", Association for Computing Machinery Inc., New York, NY, United States, 147 (2015).

16. T. Gogar, O. Hubacek, and J. Sedivy, "Deep neural networks for Web page information extraction", in L. Iliadis and I. Maglogiannis, (eds) "Artificial Intelligence Applications and Innovations: Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations", Springer, Cham, 154 (2016).
17. V. Romero, A. Fornes, E. Vida, and J. A. Sanchez, "Information extraction in handwritten marriage licenses books using the MGGI methodology", in L. A. Alexandre, J. S. Sanchez, and J. M. F. Rodrigues (eds) "Pattern Recognition and Image Analysis: Proceedings of the 8th Iberian Conference on Pattern Recognition and Image Analysis", Springer, Cham, 287 (2017).
18. E. Sassolini, A. F. Khan, M. Biffi, M. Monachini, and S. Montemagni, "Converting and Structuring a Digital Historical Dictionary of Italian: A Case Study", Proc. of the eLex 2019 Conf., Sintra, Portugal, pp. 603-621 (2019).
19. C. Reul, S. Göttel, U. Springmann, C. Wick, K. M. Würzner, and F, Puppe, "Automatic Semantic Text Tagging on Historical Lexica by Combining OCR and Typography Classification", Proc. 3rd Int. Conf. Digital Access to Textual Cultural Heritage, Brussels, Belgium, pp. 33-38 (2019).
20. A. Stein, "Preserving Semantic Information from Old Dictionaries: Linking Senses of the Altfranzosisches Worterbuch to WordNet", Proc. 12th Conf. Language Resources and Evaluation, Marseille, France, pp. 3063–3068 (2020).
21. O. Belyaev, I. Khomchenkova, J. Sinitsyna, and V. Dyachkov, "Digitizing print dictionaries using TEI: The Abaev Dictionary Project", Proc. Seventh Int. Workshop on Computational Linguistics of Uralic Languages, Syktyvkar, Russia, pp. 57-64 (2021).
22. A. Scius-Bertrand, S. Gabay, J. Janes, L. Petkovic, C. Corbieres, and T. Clerice, "The BIR database – Identifying typographic emphasis in list-like historical documents", The 6th Int. Workshop on Historical Document Imaging and Processing, Lausanne, Switzerland, pp. 37-42 (2021).
23. Dynamsoft. Recommended Scan Settings for the Best OCR Accuracy. Dynamsoft Document Imaging Blog. https://blog.dynamsoft.com/insights/scan-settings-for-best-ocr-accuracy/ [Accessed 27/03/2021].
24. Improve OCR Accuracy With Advanced Image Preprocessing. https://docparser.com/blog/improve-ocr-accuracy/ [Accessed 27/03/2021].
25. OpenCV: Changing the contrast and brightness of an image! https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html [Accessed 27/03/2021].
26. OpenCV: Smoothing Images. https://docs.opencv.org/3.4/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.html [Accessed 27/03/2021].
27. J. J. Hull, "Document image skew detection: Survey and annotated bibliography", in J. J. Hull and S. S. Taylor (eds) "Document Analysis Systems II", World Scientific, 40 (1998). doi: 10.1142/9789812797704_0003.
28. A. Papandreou and B. Gatos, "A novel skew detection technique based on vertical projections", Proc. of 2011 Int. Conf. Document Analysis and Recognition, Beijing, China, pp. 384-388 (2011). doi: 10.1109/ICDAR.2011.85.
29. W. Postl, "Detection of linear oblique structures and skew scan in digitized documents", Proc. of the Int. Conf. Pattern Recognition, Paris, France, pp. 687–689 (1986).
30. Y. Chen and J. Wang, "Skew detection and reconstruction of color-printed document images", IEICE Transactions on Information and Systems **E84-D**(8), 1018 (2001).
31. tanwirzaman/TextSkewDetectionAlgorithms. https://github.com/tanwirzaman/TextSkewDetectionAlgorithms [Accessed 28/03/2021].
32. ABBYY Help Center. Code samples https://www.ocrsdk.com/documentation/code-samples/ [Accessed 28/03/2021].
33. Tess4J. http://tess4j.sourceforge.net/ [Accessed 28/03/2021].
34. Asprise. Java OCR and Barcode Recognition. https://asprise.com/royalty-free-library/java-ocr-api-overview.html [Accessed 28/03/2021].