An Approach to Modeling Heterogeneous Resources for Information Extraction

Lei Xia, José Iria

Department of Computer Science
University of Sheffield
Sheffield, S1 4DP, UK
{l.xia, j.iria}@dcs.shef.ac.uk

Abstract

In this paper, we describe an approach that aims to model heterogeneous resources for information extraction. Document is modeled in graph representation that enables better understanding of multi-media document and its structure which ultimately could result better cross-media information extraction. We also describe our proposed algorithm that segment document -based on the document modeling approach we described in this paper.

1. Introduction

Information Extraction (IE) from text has been an important research field within natural language processing for many years (Moens and Busser, 2006). Traditionally, most IE research has been applied to selected corpora within controlled experimental environments. In many real-world corporate environments, though, the nature and complexity of corpora is very diverse. In particular, multiple document formats abound, such as OpenDocument [1], Microsoft Office's, HTML and PDF. Contrary to plain text, these formats typically carry a mixture of textual content, metadata about the text (e.g. style information), images, tables and other media objects. However, in order to minimise processing complexity, IE systems tend to simply strip documents down to their core textual format, typically sets or sequences of tokens. While this may be enough for tasks such as text categorization (Sebastiani, 2002), where state-of-the-art approaches can afford to model the text as a bag of words, such stripping down throws away layout or relational information which would otherwise provide valuable features to an IE algorithm (Shin and Doermann, 2000)(Maderlechner and Suda, 1998)(Breuel, 2003). This is particularly true with modern multimedia documents, where valuable features can sometimes be found under and across different structural contexts and media within the document.

Each document format models the document differently, making it hard for IE systems to have available an integrated view, across formats, of the text and its related metadata and media objects. Therefore, we believe that IE systems would benefit from an integrated model to represent text and other language resources coming from heterogeneous formats and media, and, consequently, of varying complexity. In this paper we propose such a modeling-based strategy, which offers the foundation of heterogeneous resources handling.

The rest of the paper is structured as follows: we start by proposing the use of graphical representations to model documents; in section 3 we present three document models for representing plain text documents, HTML, and ODF documents. Each model expands on the previous one to expand the amount of information represented. We then elaborate on the advantages of an integrated model. Finally, we conclude by discussing two applications where documents are represented according to the integrated model are used, aiming to show the advantages of the approach.

[1] http://www.odfalliance.org/

2. Graphical Representation of Resources

The data formats we have modeled (plain text, HTML and ODF) may contain complex internal structures and relations, e.g. relations between the different media elements within the document. In order to capture such complexity, we have adopted a graph data structure to represent the data, which we found to be expressive enough for our purposes. We have adopted RDF (Brickley and R V, 1999) as the formalism to model the graph representation. We view RDF as a typed graph, where an RDF class corresponds to a node type in the graph, and an RDF property to an edge type in the graph. Both RDF classes and properties are identified by unique identifiers when specifying a model. All models presented in this paper have an equivalent RDF declarative specification behind.

The adopted graphical representation of resources is characterized as follows:

- A set of documents (of multiple formats) is represented as a di-graph $G = (V, E, f, g)$;
- $V = \{v_1, v_2, ..., v_n\}$ are the typed nodes, where v_i denotes a meaningful component (e.g. a paragraph, a token, or an image). Nodes with the same content are given a unique identifier, therefore merged in the representation. The function $f : V \rightarrow T$ associates a type with the nodes;
- $E = \{e_1, e_2, ..., e_n\}$, where $e_i = (v_j, v_k), 1 \leq i \leq n, v_j, v_k \in V$, denote the relations connecting components, mostly describing structural or document flow information. The function $g : E \rightarrow T$ associates a type with the edges;
- We introduce the concept of a 'virtual edge', fundamental to the integration of models. A virtual edge e_v is a path of size two in the graph, formed by two edges 'pointing to' the same node, that is, $e_v = (v_i, v_c, v_j)$, $(v_i, v_c) \in E \wedge (v_j, v_c) \in E$. Virtual edges arise from the data due to the fact that nodes with the same content are merged, not from explicitly modeling them.

In our models, nodes hold textual contents or media objects, e.g. a word, a sequence of strings, a figure, a video clip. A paragraph node *text:p* can be associated with another *text:p* by an edge *has_preceding_paragraph* in the graph, or associated with *draw:frame* via an edge *has_image_ref*, for instance. Other common relations modeled include the relations between textual node or image node with their respective attributes, e.g., a *text:p* node associated with a *text:style-name* node containing a list of font properties.

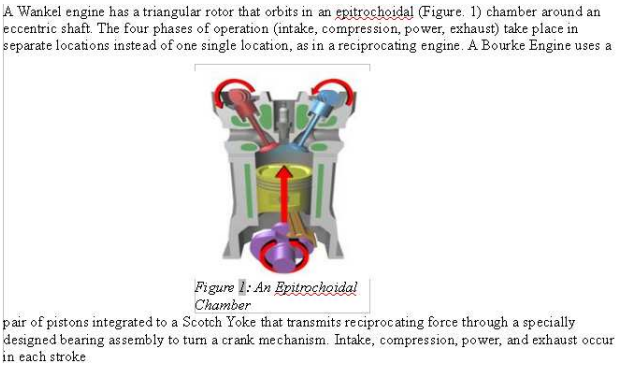
A Wankel engine has a triangular rotor that orbits in an epitrochoidal (Figure. 1) chamber around an eccentric shaft. The four phases of operation (intake, compression, power, exhaust) take place in separate locations instead of one single location, as in a reciprocating engine. A Bourke Engine uses a

Figure 1: An Epitrochoidal Chamber

pair of pistons integrated to a Scotch Yoke that transmits reciprocating force through a specially designed bearing assembly to turn a crank mechanism. Intake, compression, power, and exhaust occur in each stroke.

Figure 1: An OpenOffice Document Example

3. Modeling Heterogeneous Formats

Having defined the generic representation formalism, we turn our attention to the problem of defining the models for representing several data formats. In this paper we focus on plain text, HTML and OpenDocument formats. A basic OpenOffice writer document is shown in Figure 1, illustrating the type of documents we are dealing with. In the following subsections, we describe one model for each data format, and explore the possibility of an integrated model.

3.1. Plain Text Document Model

The plain text document model allows representing a decomposition of plain text documents into their constituent components. By 'plain text' we refer to documents that have no text styling information, no links between them and no multimedia objects attached. Plain text documents are still very commonly found in email and news groups, for instance.

Figure 2 illustrates a plain text model. In this model, the document is chunked into sentences, which are represented in the graph by the sentence nodes. Since there is no layout metadata, nodes $document\_source$ and $document$ are related by relation $has\_document$ which indicates the document belongs to a corpus. Similarly, nodes $document$ and $text\_content$ are related by edge $has\_textual\_content$. Between nodes $text\_content$ and $sentence$, we use edges $first\_sentence$ and $last\_sentence$, which only link the first sentence and the last sentence in the document, respectively. We specialise this model down to the token level. Each $token\_node$ is related by two directed edges that semantically form an undirected relation edge has_next_token and $has\_previous\_token$, which allow modeling token sequences.

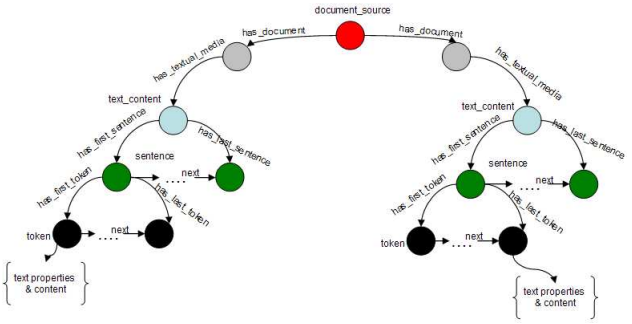

Figure 2: The plain text document model. A plain text corpora (red) is decomposed into a list documents, and each plain text document (gray) is decomposed into text sections (blue). Text sections are then tokenized (green), and additional properties such as parts-of-speech added to each token (omitted in braces)

3.2. HTML Document Model

Due to the advent of the WWW, an HTML document is probably the most common electronic resource available nowadays, making it a natural target for Information Extraction systems. Unlike plain text, an HTML document is often rich in media objects and structural layout information. This extra complexity can and should be exploited by an IE algorithm. For that purpose, we propose a model to represent HTML documents, depicted in Figure 3.

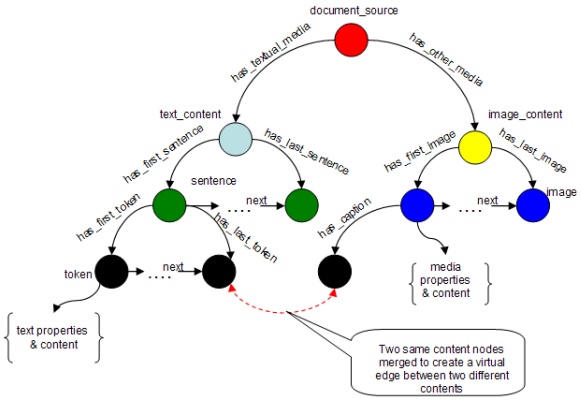

Figure 3: The HTML document model. An HTML document resource (red) is branched according to media types (blue and bright yellow nodes denote text and image content, respectively) present in the documents. Additional properties for content are also added onto each media (omitted in braces).

Compared to the plain text model, the HTML model adds a top level layer that branches according to the media types found in the set of HTML documents. The model also accommodates extra text properties, such as $font\_type$ and $font\_color$. Figure 3 also depicts the concept of virtual edge (color edge): the model connects textual entity to image by merging the shared text content node. No edge is explicitly added between such two entities in the representation, rather entities are indirectly related by the fact they share a common content node.

3.3. ODF Document Model

The OpenDocument format has become popular in recent years due to its open nature and tools compatibility with other major Office applications. Very much like HTML, ODF is based on XML, and accommodates rich multimedia content and layout information.

In order to model ODF, we expand our HTML model. The ODF model, depicted in Figure 4, adds another top layer that branches according to ODF document type, i.e. OpenDocument text, spreadsheet, and presentation. In complex IE tasks, information to be extracted is present in more than one media or document (e.g. in cross-media knowledge extraction (Ferraro et al., 2007)), therefore we need to model information across documents of different formats. For instance, with this model we can relate numerical data from an OpenDocument spreadsheet to a textual description in a OpenDocument text.

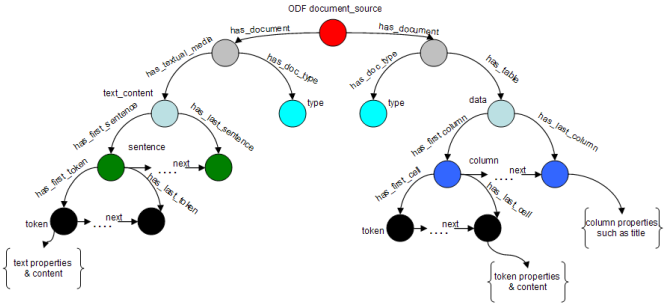

Figure 4: The ODF document model. An ODF document resource (red) is branched according to ODF document type (bright blue indicates document type). Tables are decomposed into cells (black). Properties are added to each cell to represent any styling information applied to their content.

3.4. The Advantages of an Integrated Model

In the previous sections, we have described three different document models, each able to address a specific document format. It is natural to explore the possibility of merging those models into an integrated model, where heterogeneous resources can be represented in one single graph. Such an integrated model enables IE tasks to abstract away from the underlying document format and access the content in an uniform way. For example, the integrated model enables performing IE across multiple document formats. Although differences between models lie in supported multimedia content, added layout information, it is possible to integrate models due to the fact that some node and edge types amongst these models overlap, such as token, $has\_sentence$ relation, and $part\_of\_speech$ tags. Figure 5 illustrates a simplified model, by introducing a layer of media type description (i.e. text content) for plain text model and replacing with common identifiers on these models, we obtain a model that can accommodate two different media at a time. The example illustrates the advantages of an integrated model is being able to annotate documents across media, see Figure 5.

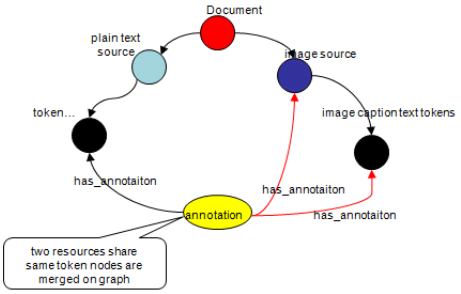

Figure 5: Cross-resource annotation. A Document (red node) containing plain text and image content which is branched according to content type (left branch for textual content and right branch for image content). Annotation (yellow node) can then be performed over different type of content, independently from the provenance of the content, and in particular its document format. Red arrows ($has\_annotation$ relation) indicates that image content can be annotated on image and its text caption, which shares the same annotation concept with textual content on left branch.

4. Document Structure Analysis Using Document Graph Model

Document processing literature discusses several approaches to extract structure information from PDF, HTML and other structured documents, see (Laender et al., 2002) for an overview. Arasu and Garcia-Molina (Arasu et al., 2003) et. al and Rosenfeld et. al. (Crescenzi et al., 2001) approaches are based templates that characterize each part of the document. These templates are either extracted manually or semi-automatically. Rosenfeld et. al. (Rosenfeld et al., 2002) implemented a learning algorithm to extract information $(author, title, date, etc.)$. They chose to ignore text content and only use features such as font, physical positioning and other graphical characteristics to provide additional context to the information.

In contrast with these previous approaches we propose a two stage document analysis algorithm working with document models which are pre-defined for ODF document format. However, it is worth to point out that these algorithms are not limited to ODF document format only. The algorithm is divided into two stages, 1). a simple analysis split the document into coarse-grain-blocks where it according to document headings and sections; 2). further analysis uses a measure of proximity to generate fine-grain-blocks of text and other nearby content types, such as image (e.g. an image should be related to a near paragraph and not to a paragraph two pages ahead). We describe these two division strategies in more details.

4.1. Coarse-Grain-Blocks

Formatting data like headings and paragraphs give meaningful information about the structure of the document and how it is organized into coarse-grain-blocks of information. These blocks are delimited by text with an explicit formatting ($\langle head \rangle$ *or* $\langle P \rangle$ *or* $\langle style \rangle$ etc.), or by text with simple formatting ($\langle bold \rangle$ *or* $\langle size \rangle$ *or* $\langle underline \rangle$ etc). The first analysis splits according to these features. A graph

walker traverses the graph where content is collected. Since styling information is referenced back to their content by graph edge, hence, -based on styling information we could group un-related text content into same group. This could be useful to identify text entities which are particularly emphasized in the document, for instance, text are wrote in **BOLD**.

4.2. Fine-Grain-Blocks

The algorithm parses the coarse-grain-block sequentially and generates a fine-grain-block containing text and associated images. This all achieved by traversing the document structure graph model we defined and populated. The sequential parsing is done with a sliding window over text generating fine-grain-blocks with a predefined size. The decision to use a sliding-window to generated fine-grain-blocks is motivated by the need to have blocks with overlapping information. This is essential to achieve a soft-segmentation of the document in case the algorithm misses the true structure of the document. The algorithm still generates some blocks that are with incorrect text-images associations in some fairly rich formatted documents. This imprecision is negligible with documents that have a more standard formatting, and the use of cross-media analysis further reduces this effect as we shall see in the next section.

4.3. Text-Images Associations

A multimedia document can express an idea across different modalities, with each text paragraph and image offering specific support to the same exposed knowledge. However, it is not straightforward to know which elements refer to the same information blocks and Cross-references (e.g. image caption) can support how each text paragraph or sentence related to each image, example include (Crescenzi et al., 2001) (Arasu et al., 2003) (Rosenfeld et al., 2002). Co-occurrence of patterns across different media types can improve the detection of the most informative text-image associations.

4.3.1. Implicit Associations

Some associations can be better delimited when one does some simple text analysis to find one implicit associations as the following example:

> ... the overheating on part X, above **diagram**, shows...

. To detect these associations some text patterns are previously encoded in the algorithm as tokens that refer to particular object types. In Implicit associations further analysis could locate the image above this text, however, to avoid missing the correct associations we pair-up several nearby images to generate the text-image associations.

4.3.2. Explicit Associations

Cross-references are an explicit way of associating a sentence or a paragraph to an image. The following example illustrates this case:

> ...the overheating on part X, see **Figure 1**, shows...

. In this case the 'Figure 1' text is generated automatically by a metadata element, thus, only through the analysis of the graph will be possible to form a meaningful pair between that sentence and the mentioned image. Note that these associations can also be present as free text for which the previous method is more adequate. Another explicit association is using cross-references in the case of figure captions. The algorithm we described above can be rather hard to apply where without a effective document structure representation. Our document graph model has demonstrated, with its support, the document and content segmentation can be much easily achieved.

5. Tools

Runes(Iria and Ciravegna, 2006) is a framework [2] for representing objects in a graph. It is possible to use runes to represent heterogeneous resources under a common representation, where relations are available for aid of IE tasks or any other purpose. In our work, Runes was employed to construct a semantic representation of the document models. Runes 1) supports RDF model as generic modeling language for our document modeling, and 2) provides a flexible plugin development environment. Graph building and content processing are build on the foundation of the Runes plug-in framework. Runes encourages small step content decomposing for improved code re-usability, hence each plugin is a small processor. A collection of plugins is used to build the graph according to the provided RDF model.

5.1. Enriched Annotation Toolkit

It is rather trivial to annotate plain text document, since annotation can be directly placed in the document without damage the original document. However, such method is not advisable in ODF document annotation without considerable effort. Therefore, we have developed a toolkit for HTML and ODF annotation. AKTiveMedia (Chakravarthy et al., 2006) is a user centric system for cross-media document enrichment; it uses semantic web and language technologies for acquiring, storing and reusing knowledge in a collaborative way, sharing it with other members of the community. It was originally developed for HTML based content. We have employed AKTiveMedia as the base system and extended its annotation capability on ODF format. To achieve that, we took advantage of an unified document model, since it allows merging same identified content node from different resources. The annotation5 is now independent from the underlying document format. In most of cases, only simple document conversation, content specific runes and an existing annotation tool are required to extend annotation task to multiple document formats. The ODF source is fully annotated on integrated model and with rich layout and multimedia information attached.

6. Conclusion

This paper has described an approach of modeling heterogeneous resources on a graph based representation for information extraction tasks, where structural and media information is appended on top of traditional textual content

[2] http://sourceforge.net/projects/runes/

as the extra dimension for IE. We presented several models, which are able to represent resources coming from different data formats. We argued that an integrated model is advantageous in several tasks, including cross-media information extraction and document annotation.

7. Acknowledgments

This work was funded by the X-Media project (urlwww.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (ITS) programmer under EC grant number ITS-FPO6-026978.

8. References

Arvind Arasu, Hector Garcia-Molina, and Stanford University. 2003. Extracting structured data from web pages. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348, New York, NY, USA. ACM.

Thomas M Breuel. 2003. Information extraction from html document by structural matching. In *Second International Wrokshop on Web Document Analysis*, pages 11–14, Edinburgh, UK.

D Brickley and Guha R V. 1999. Resource description framework (rdf) schema specification. Technical report, W3C.

A Chakravarthy, F Ciravegna, and V Lanfranchi. 2006. Cross-media document annotation and enrichment. In *1st Semantic Authoring and Annotation Workshop (SAAW2006)*, Athens GA USA.

Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. 2001. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 109–118.

Massimo Ferraro, Luca Gilardoni, Chrustian Biasuzzi, Piercarlo Slavazza, and Fabrizio Lovato. 2007. Requirement analysis for compound document processing. Technical report, Quinary Spa.

José Iria and Fabio Ciravegna. 2006. A Methodology and Tool for Representing Language Resources for Information Extraction. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, May.

A. Laender, B. Ribeiro-Neto, A. Silva, and J. Teixeira. 2002. A brief survey of web data extraction tools. In *SIGMOD Record*, volume 31, June.

Gerd Maderlechner and Peter Suda. 1998. Information extraction from document images using white space and graphics analysis. In *SSPR/SPR*, pages 468–474.

M. Moens and R. De Busser, 2006. *Information Extraction: Algorithms and Prospects in a Retrieval Context*, chapter 1. Springer, The Netherlands.

Binyamin Rosenfeld, Ronen Feldman, and Yonatan Aumann. 2002. Structural extraction from visual layout of documents. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 203–210, New York, NY, USA. ACM.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

C. Shin and D. Doermann. 2000. Classification of document page images based on visual similarity on layout structures.