Position Papers of the Federated Conference on
Computer Science and Information Systems pp. 25–30

DOI: 10.15439/2015F380
ACSIS, Vol. 6

The Serialization of Heterogeneous Documents

Peter John Hampton, William Blackburn, Hui Wang
Artificial Intelligence and Applications Research Group
Ulster University, Jordanstown
United Kingdom, BT37 0QB
Email: hampton-p1@email.ulster.ac.uk
{wt.blackburn, h.wang}@ulster.ac.uk

***Abstract*—Tasks involving the analysis of natural language are typically conducted on a corpus or corpora of plain text. However, it is rare that a document is unstructured and freeform in its entirety. Documents such as corporate disclosures, medical journals and other knowledge rich archive contain structured and loosely-structured information that can be used in a variety of important text mining tasks. In this paper we propose a syntactical preprocessing architecture to serialize presentation-oriented documents to a machine readable format that aspires to preserve the document structure, contents and metadata. We introduce a hybrid pipeline architecture, discussing the various processes and the future research direction that could potentially lead to a holistic representation of heterogeneous documents.**

I. Introduction

KNOWLEDGE mining researchers and practitioners have been implementing techniques to aid and enact decision-making from knowledge discovery tasks. However, various challenges restrict the computational understanding of language found in such documents as analysis has traditionally focused on plain text (1; 2; 3). This paper proposes a hybrid preprocessing architecture for preserving a documents contents in its entirety and converting selected entity classes to their canonical form, enabling deeper analysis.

Although there are arguably many documents of interest, we focus the attention of this paper on corporate disclosures, specifically interim financial reports (10-Qs) due to the depth of knowledge and the complexity of their composition. We demonstrate, at a high level, a multistage architecture in Fig. 1 that combines both statistical and rule based approaches for serialization to preserve the documents structure and content.

Diverse developments in Information Retrieval methodology have been made over the past two decades, which could make it sufficiently easier to represent unpredictable document formats and associated contents. We describe related work and motivations behind this research in Section II. Section III analyzes the document structures of five company interim disclosures. In Section IV we compare various mainstream data serialization formats while concluding the advantages and disadvantages among them. Section V describes the pipeline components depicted in Fig. 1 in substantive detail, breaking down each process into a set of processes. The paper concludes in Section VI which sets a future direction for our research.

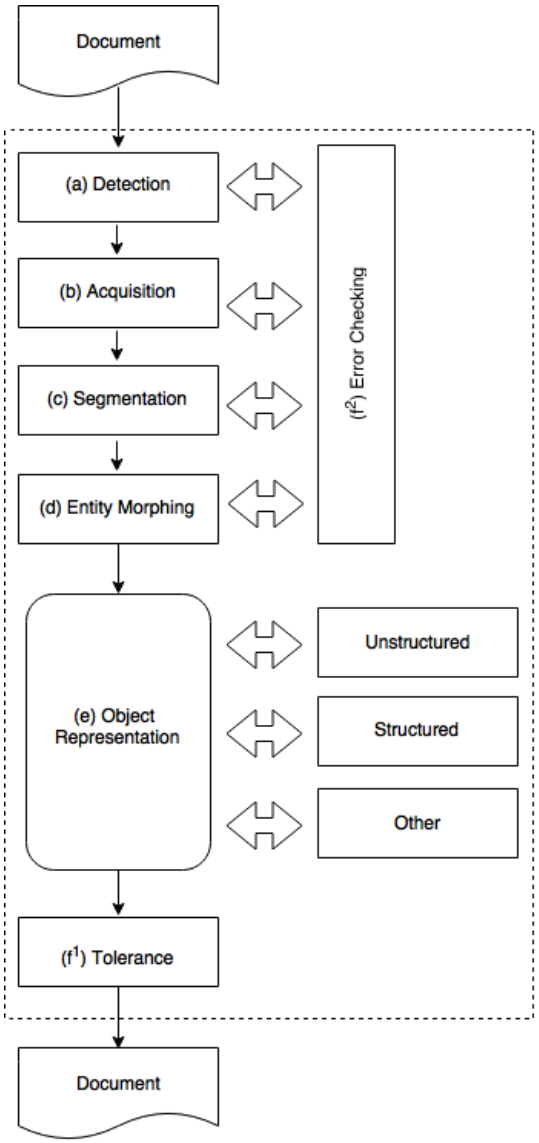

Fig. 1. The Hybrid API Architecture described in this paper for the syntactic representation of heterogeneous documents. At a high level, the depicted analysis pipeline is broken down into 6 stages: Detection (a), Acquisition (b), Segmentation (c), Entity Morphing (d), A cache object for document representation (e) and a duel error handling module (f). A document is introduced into the pipeline and subject to numerous decomposition stages.

II. Related Work

Early attempts at analyzing structured data in heterogeneous documents include the Douglas *et al* study (4) which represented tabular data using spaces in plain text. These researchers presented an analysis of table layout and the associated linguistic characteristics. However, wider linguistic complexities are presented in plain text representation of tables. This can include complex header parsing, redundant or null cell representation and wider linguistic complexities. We aim to extend their problems of characteristic syntax and apply a set of transformations to the contents.

More recent work includes the Clark & Divvala research (5) which aims to discover a holistic view of the document, achieving a deeper *semantic* understanding of articles such as academic computer science papers by placing emphasis on figures such as charts. Likewise, the data found in corporate disclosures tend to be multifaceted in nature, that being a mix of unstructured, loosely-structured, unstructured text and miscellaneous data. Their open-source solution agnostically parses a document and locates the areas wherein figures or tables could reside by reasoning about the empty regions within that text, achieving success due to it's relaxed formatting assumption.

III. Background & Analysis

In this paper we refer to freeform text, that is sentences with no predictable syntactical structure as *unstructured* content. In turn, we refer to information in tabular form as *structured data*. Although the term 'loosely-structured data' is typically used to refer self-describing data (6; 7), we use it in this paper as a means of classifying information that is not presented in a structured or unstructured form. This can take the form of bullet points, footnotes, images etc.

We analyzed a random sample of five interim statements published by different software companies between July and December 2014 as listed on the British Alternative Investment Market (AIM). The results portrayed in Fig. 2 show that although the majority of content is unstructured in nature, the sample set had structured and loosely-structured data accounting for 22% to 65% in terms of token count.

Fig. 3 on the other hand shows that although unstructured, free form text and miscellaneous data increased with page count, structured and loosely-structured data could be considered random and unpredictable. The share of structured and loosely-structured information is, in our opinion, substantial and shouldn't be left out of text mining tasks due to the risk in knowledge loss. We propose syntactically serializing the document in a machine readable format whose schema is flexible enough to adapt to unpredictable content and volatile formatting. It is clear however from this preliminary analysis of the small data set that if it is possible to serialize documents, a format would require several characteristics to make this possible such as flexibility, system interoperability, etc. We review multiple data serialization formats in the next section and select the most appropriate for serializing multifaceted documents written by specialist humans.

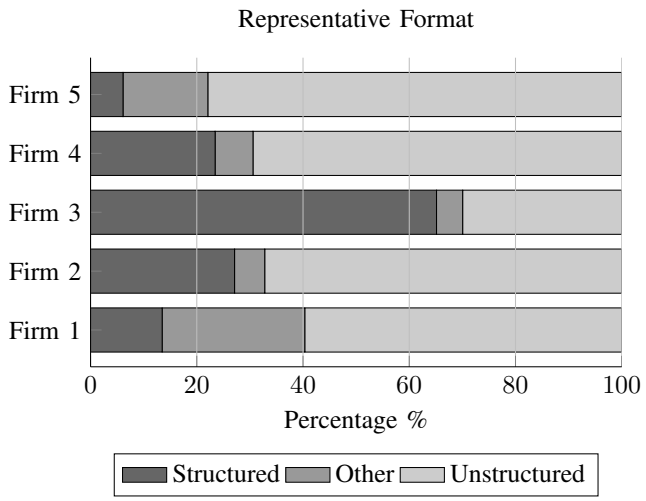

Fig. 2. Five Interim statements broken down to their representative cluster. This initial study found there was no predictable cluster share of the documents studied.

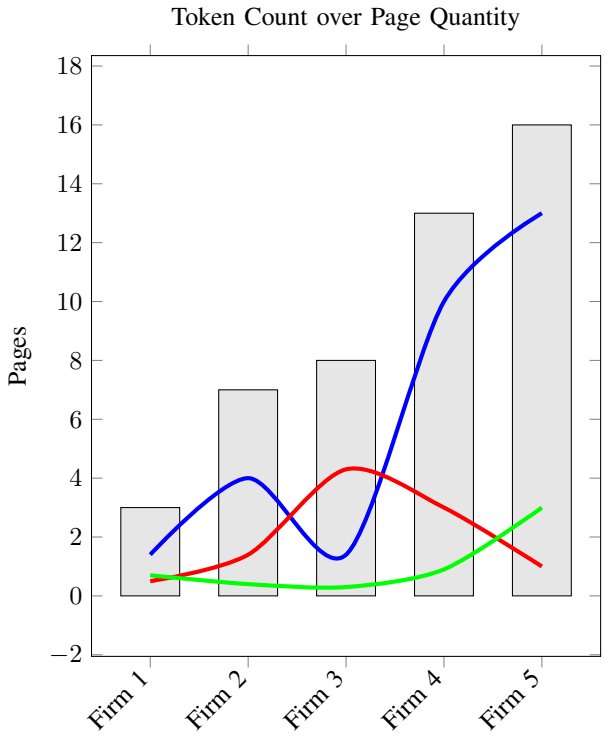

Fig. 3. The change in token count over page count. The results show that although unstructured content increases with page count, it can be deemed unpredictable the share of structured, loosely structured and other miscellaneous data will represent of the over all document. The blue line shows the change in unstructured data wheras The red line shows changes in structured data and The green line shows the changes in other data formats.

TABLE I
THE DATA TYPES OF JAVASCRIPT OBJECT NOTATION

Data Types	Example	Description
Numeric	1, -5, 0.9, 34543987584	Can represent a signed integer or floating-point number and may use exponential E notation.
String	“Jack”, “Jill”	A sequence of characters. This data type supports 8, 16 and 32 bit unicode transmission formats.
Boolean	true or false	Either True or False values.
Array	[0, “Jill”, [1, 2], null]	An ordered list of all the data types in this table including null.
Object	“names”{ “name”: “Jack”}	An unordered set of key-value pairs, where the keys must be specified explicitly in a String format.
None Type	null	A vacant value often used as a placeholder.

IV. SERIALIZATION FORMATS

Data Serialization is the process of translating data structures into a sortable format to be loosely reconstructed at a later time if required. We focus our attention on formats that are data oriented rather than document markup oriented. In the upcoming subsections we review XML, JSON and YAML.

A. XML

XML (eXtensible Markup Language) is probably one of the most popular document serialization formats since the rise of Web 2.0, a milestone in web evolution that democratized content, converting a generation of information consumers into content creators. XML is described by the World Wide Web Consortium as a flexible user specified markup scheme, whose elements are not subject to formatting rules (7). One advantage XML has is that it is human readable and can be converted to various serialization formats using XLST (eXtensible Stylesheet Language Transformation) or an independent parser.

B. JSON

JSON (JavaScript Object Notation) is a relatively newer serialization format compared to XML, which is lightweight, thus lower processing overhead (8). It is regarded as a lightweight alternative to XML, which can preserve the native data type of the selected entity and can be used for server parsing (9; 10). We describe an overview of JSON data types in Table I.

C. YAML

YAML (YAML Ain’t Markup Language) is a superset of JSON that offers an alternative, user friendly syntax by replacing various nested delimiters, such as list braces, object colloquially and quote marks with whitespace. YAML supports a powerful feature that we believe could benefit many natural language processing researchers called anchoring, which enables embedded object referencing and can handle relational information similar to a traditional SQL database (11).

D. Discussion

Due to its popularity, XML seems an attractive choice. However to query an XML, XPATH (XML Path Language) is required to parse the document and would be considered inefficient compared to newer standards. Further we believe it could prove difficult to train a machine to autonomously parse the semi-structured nature of XML document. Adopting JSON on the other hand, a specified path could be explicitly declared or discovered with ease due to the tree like structure. After reviewing numerous studies and experimentation, we chose to rule out YAML in it’s current state[1] as it typically requires an external parser and we found that when automatically generating YAML documents there was possible corruption. However, future experimentation with YAML could yield an interesting study as it is much more verbose and is being used in various innovative ways.

JSON compared to XML has a steeper learning curve and debatably a much more complex syntax. The comprehensive Eriksson-Halberg study surveyed performance and the syntax of JSON and YAML and found JSON implementations to be ‘*many times faster than YAML for both serialization (dumping) and deserialization (loading)*.(12)’ The researchers concluded that the complexity behind processing YAML formats to be the reason for this. We feel that due to the level of expressivity and efficient performance of JSON that we should adopt it, for the current time being, to represent document inputted to our pipeline.

V. PIPELINE ARCHITECTURE

In Fig. 1, we showcased and briefly discussed our high-level pipeline architecture for serializing documents with an unpredictable structure and format at a high level. Here, we describe the various stages an inputted document goes through to be serialized. As corporate disclosures, medical journals and academic papers are commonly published in Portable Document Format (PDF), we describe a PDF as the input to this propositional system.

A. Detection

The document format is first determined by the MIME type (Multipurpose Internet Mail Type). This phase inserts the content type header along with the document to the Acquisition phase (b) for the documents conversion and decomposition.

B. Acquisition

Disclosures released by a company are often published in Portable Document Format (PDF). This *presentation over*

[1]At the time of publication, the latest version of YAML was v1.2

content oriented format presents a number of problems for natural language acquisition and analysis. Further, the document's structure is unpredictable and can change between structured, loosely-structured or free form texts at any time. Therefore, we first propose an introduction stage dedicated to the decomposition of the document, which we call the *Acquisition and Segmentation* phase of our pipeline shown in Fig 4.

Once the pipeline has accepted the document, two concurrent processes must take place. The first process aims to extract the document metadata. This metadata can include information such as creation date, modified date, author name(s), publisher geo-coordinates and other miscellaneous information (13).

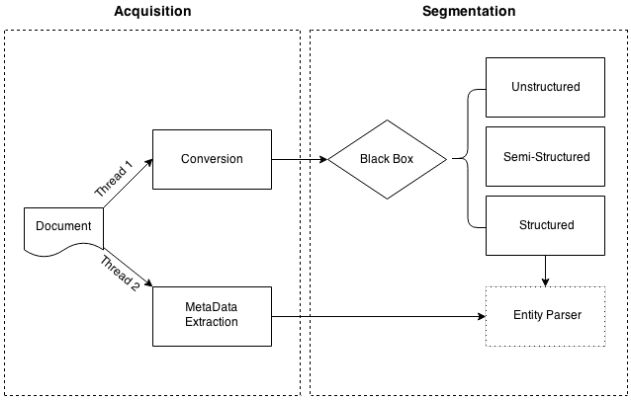

Fig. 4. The Acquisition Process: The detected document format is accepted bu the pipeline. One process extracts the metadata (if any) and the second process clusters the document into one of three categories.

C. Segmentation

The segmentation process converts the document into an HTML (HyperText Markup Language) format, assuming the content type header is not already in HTML format, preserving bullet points and table formats, the latter for example by placing null values in redundant cells. The source document is first converted to HTML in order to traverse semi-structured nodes in the document. To categorize the unstructured, loosely-structured, structured and other miscellaneous information, we propose a black box clustering mechanism to classify innumerable sections of the document that the set of token sequences belong to. During this phase, we also assign various reference data to the section, which includes the page number, the section number, the paragraph index, and index of the sentence in that paragraph, to somewhat preserve the presentational elements of the document.

D. Entity Morphing

Due to the unpredictable and creative use of language within corporate disclosures, the need to manipulate tokens became apparent at early stages of this research. This component described is a series of objects that manipulate text into a common reusable and storable format in the form of objects. We narrow the research direction to seven named entity types, which can be efficiently remodeled using pattern matching techniques such as regular expressions. These entity types are *monetary values, percent, decimals, durations, times, dates and miscellaneous quantity values* with examples provided in Fig 5.

$(\$1.1b \rightarrow \{'\$', 1100000000\})$
$(\$4.7million \rightarrow \{`\$', 4700000\})$
$(6.3p \rightarrow \{'GBP', 0.063\})$
$(12thMarch2015 \rightarrow \{'20150312000000'\})$
$(€66,000,000 \rightarrow \{'€', 66000000\})$
$(5.2\% \rightarrow \{0.052\})$
$(-2perc \rightarrow \{-0.02\})$

Fig. 5. Parsed Tokens: examples depicting an input x and output y where (.) represents an input to the annotator object's helper function. $(x) \rightarrow y$

We advocate the removal of PERCENT objects by converting all percentages found in the text to a decimal format. It was found that performing calculations on the extracted entities was possible if presented in a decimal format, a technique we plan to explore further in future research.

E. Object Representation

We refer to unstructured content as content that has no predefined data-model and has an unpredictable structure, often mass text that can contain various objects such as money, time, quantities, and so on. We serialize the sentences into an array of JSON objects, maintaining the various indexes discussed in Section A of the pipeline. We use the Punkt Tokenizer in the NLTK as described by Bird (13) which is a model trained using a unsupervised machine learning algorithm for sentence boundary detection. However, various popular tokenizers could also prove appropriate for this task such as the Stanford Tokenizer or TrTok which prove effective when parsing messy web text data. We provide a serialization example of an unstructured paragraph in 5.1.

Example 5.1 (Unstructured Paragraph Example):

```
{
    "sentences": [{
        "paragraph_index": 1,
        "sentence_index": 1,
        "cluster_index": 1,
        "page_index": 1,
        "sentence": "This is a sentence."
        },
        {
        "paragraph_index": 1,
        "sentence_index": 2,
        "cluster_index": 1,
        "page_index": 1,
        "sentence": "...also a sentence."
    }]
}
```

Khusro *et a*l (15) note that the *detection, extraction and annotation* of tables within heterogeneous documents have been quite a significant research problem in Information Retrieval for many years. The structured serialization stage of the hybrid pipeline aims to serialize complex and unpredictable tables into JSON format. Tables are information rich data stores, which contain a lot, often audited factual or objective information. We show how a parsed table (Table II) would be serialized in JSON in Example 5.2. This is possible using a very carefully programmed set of rules. This white box approach has proved effective for our small sample of documents but may need to be extended with intelligent based processing as our architecture scales and formats become increasingly complex.

TABLE II
FDP.L PROFIT & LOSS EXCERPT DATED NOVEMBER 2014

	6 Months Ended, 31 August 2014	**6 Months Ended, 31 August 2013**
	£'000	**£'000**
Revenue	37,507	34,381
Cost of Sales	(27, 606)	(25,313)
Gross Profit	9,900	9,068

Example 5.2 (FDP.L Table Serialized):

```
{
    "profit_loss": {
        "revenue": {
            "2014-08": 37507000,
            "2013-08": 34381000
        },
        "cost_of_sales": {
            "2014-08": -27606000,
            "2013-08": -25313000
        },
        "gross_project": {
            "2014-08": 9900000,
            "2013-08": 9068000
        }
    }
}
```

F. Error Checking & Tolerance

We implement two logic based error checking modules depicted in Fig. 1 as f_1 and f_2 respectively. The error handling processes verifies the accuracy and corrects mistakes made by the various concurrent processes. Scenarios covered by the error checker include:

- The document format has been correctly identified.
- Verify an objects reference data.
- Cast string named entities to their native format.
- Testing the structure of the tree-like output.

VI. FUTURE WORK

In this initial study we have proposed a novel architecture to serialize and represent a document and its contents for further analysis. We feel there is still scope for vast improvement and research into converting presentation oriented documents into a machine readable format that a human can easily debug. First, with the rise of the semantic web and linked data, we believe that extending our serialization model to JSON-LD (JavaScript Object Notation for Linked Data) may be appropriate. Secondly, there is potential to cluster knowledge together as in *Example 6.1* to solve various interoperability and distribution problems within Big Data systems.

Example 6.1 (Multiple Sources Serialized):

```
{
    "id": 1,
    "name": "Company X",
    "collection": {
        "disclosures": { ... },
        "social_media": { ... },
        "company_news": { ... },
        "media_news": { ... },
        "stock_price": { ... }
    }
}
```

Finally, many messages published over the web are transmitted natively in a JSON format and could prove appropriate to build up a profile of different sources into a single object for efficiency. We believe from this early study that this would make Big Data analysis on these documents much more efficient and manageable (15).

REFERENCES

[1] Comeau, D. C., Liu, H., Doğan, R. I., & Wilbur, W. J. (2014). Natural language processing pipelines to annotate BioC collections with an application to the NCBI disease corpus.

[2] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 55-60).

[3] Liu, M., Xu, W., Ran, Q., & Li, Y. (2015). Using Natural Language Processing Technology to Analyze Teachers' Written Feedback on Chinese Students' English Essays.

[4] Douglas, S., Hurst, M., & Quinn, D. (1995). Using natural language processing for identifying and interpreting tables in plain text. In Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (pp. 535-546).

[5] Clark, C., & Divvala, S. Looking Beyond Text: Extracting Figures, Tables and Captions from Computer Science Papers.

[6] Ding, L., Zhou, L., Finin, T., & Joshi, A. (2005). How the semantic web is being used: An analysis of foaf documents. In System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on(pp. 113c-113c). IEEE.

[7] Li, X., Li, F., & Chen, X. (2015, April). Distributed GIS framework design based on XML and Web Service. In 2015 International Conference on Intelligent Systems Research and Mechatronics Engineering. Atlantis Press.

[8] Hwang, C. G., Yoon, C. P., & Lee, D. (2015). Exchange of Data for Big Data in Hybrid Cloud Environment.

[9] Niu, Z., Yang, C., & Zhang, Y. (2014). A design of cross-terminal web system based on JSON and REST. In Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on (pp. 904-907). IEEE.

[10] Smith, B. (2015). Creating JSON. In Beginning JSON (pp. 49-67). Apress.

[11] Ben-Kiki, O., Evans, C., & Ingerson, B. (2005). YAML Ain't Markup Language (YAML^{TM}) Version 1.1. yaml. org, Tech. Rep.

[12] Eriksson, M., & Hallberg, V. (2011). Comparison between JSON and YAML for data serialization. The School of Computer Science and Engineering Royal Institute of Technology.

[13] Bird, S. (2006). NLTK: the natural language toolkit. In Proceedings of the COLING/ACL on Interactive presentation sessions (pp. 69-72). Association for Computational Linguistics.

[14] Smutz, C., & Stavrou, A. (2012, December). Malicious PDF detection using metadata and structural features. In Proceedings of the 28th Annual Computer Security Applications Conference (pp. 239-248). ACM.

[15] Khusro, S., Latif, A., & Ullah, I. (2014). On methods and tools of table detection, extraction and annotation in PDF documents. Journal of Information Science.