# Content enrichment with expressive document modelling to leverage the understanding of unstructured data

*Ganesh* Selvaraj*, *Karla* Taboada, *Eloy* Gonzales, and *Habib* Baluwala

All in All Analytics Limited, New Zealand

**Abstract.** Most information in an enterprise is in the form of unstructured data which is usually managed using a document database. One of the key challenges is to define a generalized data model for this unstructured data and any information extracted from it using content enrichment algorithms. It is more challenging to incorporate provenance and temporal capabilities to such data models. Semantic databases use ontologies such as PROV-O to represent their provenance information expressively, and relational databases use for example Slowly Changing Dimensions (SCDs) concepts to represent temporal information. In this paper, we present a document model which has features inspired from Dublin core, PROV-O and temporal methodologies to generalize information extracted from unstructured data using content enrichment algorithms. Provenance information enables comparison of enrichment models, allows reproducibility and facilitates complex filtering on the enriched data. Temporal metadata helps in versioning the document and enables point-in-time and history queries conveniently.

## 1 Introduction

Most of the valuable enterprise data is either unstructured or semi structured [4, 8]. This data could be conversations, emails, notes, documents, logs, images, videos, audios, etc. From this data, content enrichment algorithms are used to extract information and features that have calculable value to the organization. Few of these algorithms are identification of parts of speech, entities (e.g. persons, organizations, locations, emails, URLs, dates, currencies, etc.), sentiments, keywords and relationships between entities. Complex data matching and identity resolution techniques can link the extracted information to the enterprise structured data. One of the key challenges here has been representing our content enrichment algorithm results efficiently in a document database using a generalized document model with temporal and provenance features. Using this generalized model, the application clients can use a single interface to parse all types of document. The document model concepts we used were inspired from Dublin Core metadata initiative [14].

* Corresponding author: allinallanalytics@gmail.com

Our method for provenance has been influenced by PROV-JSON [10] to some extent and on top of that we have tried to decouple data from provenance as much as possible.Data provenance and reproducibility of computations play vital role to achieve quality analytics. Real-time data, continuous or sliced snapshot data contains some temporal aspects of it [13]. Several researchers have focused on introducing time-based document models [12, 9]. However, most of the research has centred on relational and object-oriented models.

Incorporation of temporal and provenance features enabled us to perform model comparisons, complex filtration of data using trust as a parameter, temporal information retrieval, etc.Provenance also helps in achieving reproducible process and improves data exploration [5, 7].

## 2 Content Enrichment Pipeline Overview

Natural Language Processing (NLP) algorithms have enabled us to extract entities, topics, keywords, sentiments, relationships, etc. from the unstructured data e.g. text. Our enrichment pipeline helps us to transform the unstructured data into a valuable asset and interpret the data with less human intervention.

The proposed content enrichment pipeline deals with textual formats, audio files (e.g. call center conversations) and images (e.g. scanned documents) as of now. By integrating this content enrichment pipeline with an expressive document data model (discussed in section 3), understanding and using analyses results is made simpler, intuitive, and most importantly represented the concepts of domain knowledge to a well known common knowledge.

### 2.1 System Design

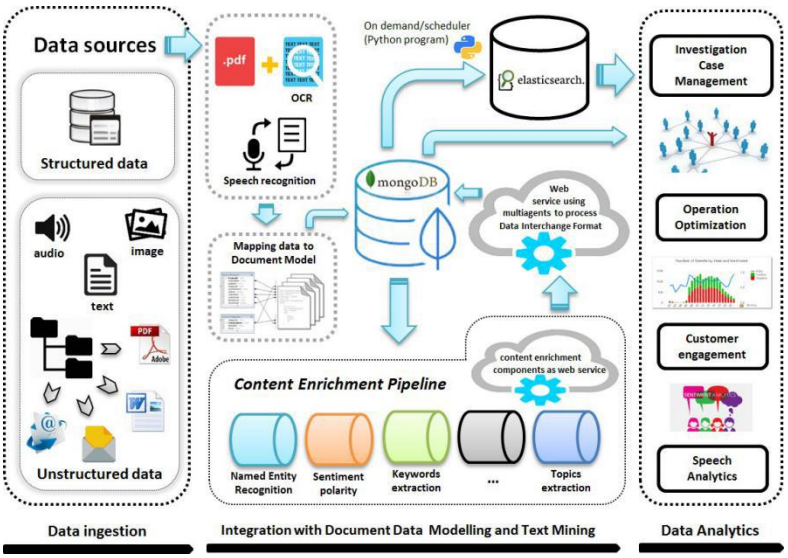


**Fig. 1.** Proposed System Architecture.

(i) **Data Collection:** The data collection is carried out in different data sources such as relational databases (e.g. Oracle), text files, emails, images, audio etc. The data collection process imports data from these sources and pass the data in a text format,

i.e. audio is converted to text using speech translation techniques (e.g. CMUSphinx software[1]), images are processed by OCR tools(e.g. tesseract[2]). This data is then sent to a document modelling engine which is discussed in the next section.

(ii) **Mapping Data to Document Model:** The document model under discussion in this paper is JSON document format, and is schema free in nature. The reason for choosing JSON is to be aligned with the format accepted by popular document databases, in our case we are using MongoDB[3]. Mapping data to document model has two processes; initial import of document and adding enrichment metadata to the document. During initial import of documents, the modelling engine is made aware of the domain we are dealing with using configuration and mapping files. If the same document that is already present in MongoDB is passed through this engine, then it performs a logical merge of the documents in-memory, and inserts a new document with a different version.

(iii) **MongoDB:** The results from the content enrichment pipeline is persisted in MongoDB (version 3.4) using a document model discussed in section 3. MongoDB is our integration point i.e. after the documents are initially loaded into it, all applications read and write to MongoDB only. Having a database as an integration point also decoupled any contradicting technologies used in our applications e.g. JDK mismatches.

(iv) **Content Enrichment Pipeline:** In this section, we explain the different components we use to extract metadata from the content.

- Named entity recognition (NER) & Relationship Extraction: It is the process of identifying entities and relationships from text data. Our primary NER platform is General Architecture for Text Engineering (GATE version 8) [6]. Some of the entities are very domain specific which are not recognized by generic domain NER's, in those cases we use regular expression patterns, JAPE[4], etc.
- Gender Identification: For a given name (either available or extracted using NER), we identify the gender e.g. male or female. Stanford coreNLP (gender annotation for English names) and dictionaries of male and females names in English, Maori, Chinese, Indian, Spanish among others are used. Currently, dictionary approach is used if Stanford core is unable to identify gender.
- Sentiment Polarity Classification: It is the process of identifying the view or opinion associated to an information. We use Stanford CoreNLP libraries [11], which is based on a new type of Recursive Neural Network that builds on top of grammatical structures. Keyword Extraction: It is the process of extracting keywords (both generic and domain specific) from the unstructured data. We are using Knime's Keygraph (graph-based approach) and Chi-square keyword (co-occurrence statistics) extractor.
- Topic Extraction: It is the process of identifying topics a document could be related to. We are using Knime's Topic Extractor (Parallel LDA (Latent Dirichlet Allocation) statistical model [3]) node. The input is a collection of documents and the output is the document collection with topic assignments and the probability for each document to a certain topic.

(v) **Data Interchange Format to Document Model:** The content enrichment pipeline and its components can use different technologies, tools, programming languages, etc. Each

[1] https://cmusphinx.github.io/

[2] https://github.com/tesseract-ocr

[3] https://www.mongodb.com/

[4] https://gate.ac.uk/sale/tao/splitch8.html

of these technologies, for the same algorithm provides the output in a different format, and we do not want to implement our relatively complex document model mapping logic into all these components. So we came up with a lightweight data interchange format (DIF), which all these components will use to represent and exchange their results. We have a document modelling web service, which coordinates communication from the content enrichment components to MongoDB. This web service receives the DIF and converts it to the corresponding document model.

## 3 Proposed Document Model

A document can be defined as a collection of information (either core and/or meta) about something or even several things usually from a domain. This core or meta information can be either structured, semi-structured or unstructured. We use the content enrichment pipeline discussed earlier to extract more information from the unstructured or semi-structured part of the document. Our document model can handle unstructured data, other associated semi- structured metadata available from the domain (e.g. from legacy relational databases), and the extracted data from the enrichment pipeline. Along with this data, the document model features a temporal and provenance model as well. Documents are expressed in JSON as per requirement for document databases, hence the document model is also expressed in JSON to have a faithful representation.

### 3.1 Document Components

In this section, we describe the different components of our document model.

(i) **docId:** Every document will have an identifier in the form of a universal resource identifier (URI). This URI is unique to a document, but across versions of the same document, this is not unique. Each document has a version number, and along with this docId, this will create a compound key for unique identification of a document. If the base URI is like this http://www.xyz.nz/, then the document from domain retail, with id 1 will have an URI such as http://www.xyz.nz/retail?id=1. The domain is used as one of the fragments in the URI to prevent overlap of same identifiers across two domains.

(ii) **version:** Every document will have version information, which is of type number.

(iii) **textContent:** Any unstructured data of the document is stored in this field. When the textContent is cleansed e.g. boilerplate removal, the original text is retained and the cleansed text is added to the ”cleansedTextContents” field.

(iv) **sentiments:** The sentiment value associated with a document is stored in this field. It will contain one of these values - very negative, negative, neutral, positive, very positive. Keywords extracted from the document also follows a similar model to the sentiment model.

(v) **parties:** Person or organization along with their roles e.g. customer, agent extracted from the text is generalized and stored in a field called ”party”. Based on the specialization, a party will have more or fewer fields. We use the ”type” field to classify the party.

(vi) **names:** Any name (belonging to person or organization) mentioned/extracted from the text is stored in this field. In our current enrichment pipeline, we detect the gender according to the name, hence the gender model is also placed alongside the name model.

(vii) **addresses:** Any address information mentioned/extracted from the text is stored in this field. Each component of address can have coordinates information.

(viii) **topics:** The results of topic modelling are stored in this field. The "topicProbability" field contains the probability of the topic for that document. TopicWords has an array of pairs -"topicWord" and "topicWordWeightage" which are generated during topic modelling. Topic modelling is performed in a collection of documents, so whenever a topic modelling is performed, a corpusId is generated.

(ix) **metadate:** Date and time information is captured using "metadate" field. These meta dates can be attached to any part of the document.

(x) **prov:** Prov is a short hand for provenance. Provenance represents information such as who, how and when about a change. The prov model used in our document model is based on Prov-O ontological model [2]. Prov-O expressed the provenance using three concepts: entity, activity and agents. When modelling with documents, ideally any change will have only one prov, but a prov can have any number of activities. Nested activity can mean its a chain reaction (i.e. an activity influenced other). Adjacent activity means several activities (different ones) have said the same thing. The docId for prov and activities are randomly generated unique Id's to help transactional systems to easily locate a piece of information.

(xi) **deleted:** A boolean field which says if a document is deleted. We do not use physical delete, hence having a delete field to represent the deleted status of the document. By default deleted is not added to a document, and it is only added when explicitly specified.

(xii) **relationships:** Even though document model is not the most ideal for representing relationships, but we wanted to persist this information so when we build graphs later, we could make use of this information. Any relationships extracted from the unstructured data is represented using index free adjacency methodology, i.e. whenever a relationship is found for a document (e.g. party), a relationship document model is added for that part of the document. Every document and any embedded document has a docId field, and the relationship model makes use of this docId to refer the target document.

## 4 Example Queries

In this section, the expressiveness and usability of the proposed document model is described with help of few example queries. Consumer Compliant Dataset [1] has been used in this paper. It is a public dataset containing complaints received about financial products and services published by the US Consumer Financial Protection Bureau.

### 4.1 Information Retrieval using Queries

MongoDB is used to persist the dataset in our document model. For this experiment, a Mongo collection called "CIE" is created. The effectiveness of the proposed document model is described in this section, especially referred to write expressive queries using the MongoDB query language are used.

(i) **Query 1:** The query shown in Listing 1 finds all documents which are affected by any entity extraction algorithm. This is a very useful audit/profiling query to discover which documents have been used or missed out in NER process.

```
db.getCollection('CIE').find({"parties.prov.activity": {$elemMatch: {"name": {
    $in: ['EntityExtraction']    }}}})
```

**Listing 1.** Query 1

(ii) **Query 2:** A common use case in machine learning is the evaluation or comparison of results by different algorithms. The proposed document model allows for comparison of results from different methods, e.g. in Listing 2 finds all documents with different sentiment values i.e. same document with negative and positive sentiment.

```
db.getCollection('CIE').find({"sentiments": {$elemMatch: {"sentiment": {$in: ['
    NEGATIVE']}}},"sentiments": {$elemMatch: {"sentiment": {$in: ['POSITIVE
    ']}}}})
```

**Listing 2.** Query 2

(iii) **Query 3:** A more complex example is shown in Listing 3; a query to find all documents with negative sentiment and which have the key word extracted as *"child"*. From an operations point of view, a document involving a child and negative sentiment is a priority to look into it, and with our document model, such expressive queries can also be easily represented.

```
db.getCollection('CIE').find({$and:[ {"sentiments": {$elemMatch: {"sentiment": {
    $in: ['NEGATIVE']}}}},{"keywords": {$elemMatch: {"keyword": {$in: ['child
    ']}}}}]})
```

**Listing 3.** Query 3

## 5 Conclusion

In this paper, we presented a document model to capture the enrichment results along with provenance and temporal metadata. The provenance feature based on PROV-O is added to the document model which enables content enrichment model comparison and reproducibility. Temporal capability in the document model allows for point in time and history queries. The document model is generalized for the most of the concepts. Hence applications can use a single interface to consume the document model for multiple domains as well. Our content enrichment pipeline comprises of different (and sometimes opposing) technologies, methodologies and tools, but they all work together by using a document database (MongoDB) as the integration point.

## References

1. Consumer financial protection bureau. https://catalog.data.gov/dataset/consumer-complaint-database, [Online; accessed 12-May-2017]
2. Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J.: Provo: The prov ontology. w3c recommendation. Online]. http://www. w3. org/TR/provo (2013)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research 3(Jan), 993–1022 (2003)
4. Blumberg, R., Atre, S.: The problem with unstructured data. Dm Review 13(42-49), 62 (2003)
5. Cheah, Y.W., Plale, B.: Provenance analysis: Towards quality provenance. In: E-Science (e-Science), 2012 IEEE 8th International Conference on. pp. 1–8. IEEE (2012)

6. Cunningham, H., Wilks, Y., Gaizauskas, R.J.: Gate: a general architecture for text engineering. In: Proceedings of the 16th conference on Computational linguistics-Volume 2. pp. 1057–1060. Association for Computational Linguistics (1996)
7. Davidson, S.B., Freire, J.: Provenance and scientific workflows: challenges and opportunities. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 1345–1350. ACM (2008)
8. Feldman, R., Sanger, J.: The text mining handbook: advanced approaches in analyzing unstructured data. Cambridge university press (2007)
9. Gregersen, H., Jensen, C.S.: Temporal entity-relationship models-a survey. IEEE Transactions on knowledge and data engineering 11(3), 464–497 (1999)
10. Huynh, T.D., Jewell, M.O., Sezavar Keshavarz, A., Michaelides, D.T., Yang, H., Moreau, L.: The prov-json serialization (2013)
11. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit. In: ACL (System Demonstrations). pp. 55–60 (2014)
12. Ozsoyoglu, G., Snodgrass, R.T.: Temporal and real-time databases: A survey. IEEE Transactions on Knowledge and Data Engineering 7(4), 513–532 (1995)
13. Panigati, E., Schreiber, F.A., Zaniolo, C.: Data streams and data stream management systems and languages. In: Data Management in Pervasive Systems, pp. 93–111. Springer (2015)
14. Weibel, S., Kunze, J., Lagoze, C., Wolf, M.: Dublin core metadata for resource discovery. Tech. rep. (1998)