# Understanding and representing the semantics of large structured documents

Muhammad Mahbubur Rahman and Tim Finin

University of Maryland, Baltimore County, Baltimore MD 21250, USA
{mrahman1,finin}@umbc.edu

**Abstract.** Understanding large, structured documents like scholarly articles, requests for proposals or business reports is a complex and difficult task. It involves discovering a document's overall purpose and subject(s), understanding the function and meaning of its sections and subsections, and extracting low level entities and facts about them. In this research, we present a deep learning based document ontology to capture the general purpose semantic structure and domain specific semantic concepts from a large number of academic articles and business documents. The ontology is able to describe different functional parts of a document, which can be used to enhance semantic indexing for a better understanding by human beings and machines. We evaluate our models through extensive experiments on datasets of scholarly articles from *arXiv* and *Request for Proposal* documents.

**Keywords:** Document Ontology · Deep Learning · Semantic Annotation.

## 1 Introduction

Understanding the semantic structure of large multi-themed documents is a challenging task because these documents are composed of a variety of functional sections discussing diverse topics. Some documents may have a table of contents, whereas others may not. Even if a table of contents is present, mapping it across the document is not a straightforward process. Section and subsection headers may or may not be present in the table of contents and if they are present, they are often inconsistent across documents even within the same vertical domain.

Identifying a semantic organization of sections, subsections and sub-subsections of documents across all vertical domains is not the same. For example, a business document has a completely different structure from a user manual. Scholarly research articles from different disciplines, such as computer science and social science, may have different structures. For example, social science articles usually have *methodology* sections whereas computer science articles often have *approach* sections. Semantically these two section types share the same purpose and function, even though their details may be quite different.

Our objective is to develop and use a document ontology to describe different functional parts of academic and business documents. For example, the

*introduction* section of a research paper describes the problem statement, scope and context by explaining the significance of the research challenge. The *results* section presents and illustrates research findings with the help of experiments, graphs and tables. Finally, the *conclusion* section typically restates the paper's contribution and the most important ideas that support the main argument of the paper.

Creating such an ontology involves significant human understanding and analysis of a large number of documents from any vertical domain. It is also requires a common understanding of the structure of information presented in those documents. The common concepts across all documents should be clearly visible to the ontology developers. The developers should also understand the hierarchy of the sections, subsection and sub-subsections of a document. Hence the process to get each relationship among different concepts of a document is time consuming. Moreover, some concepts may be overlooked while analyzing the documents.

We have developed a deep learning based system to automatically determine ontology concepts and properties from a large number of documents of the same vertical domain. Our approaches are powerful, yet simple, to capture the most important semantic concepts from academic articles and *request for proposal* (RFP) documents. In the course of out this work, we experimented with and evaluated several state of the art technologies, including Variational Autoencoders (VAE) [14], Convolutional Autoencoders (CAE) [13, 27] and LDA [4].

The ontology can be used for annotating different sections of a document, which helps to understand its semantic structure. It can also be useful for comprehending and modeling types and subtypes of documents. The results can enable the reuse of domain knowledge along with text analysis, content based question answering and semantic document indexing.

## 2 Related Work

Over the last few years, several ontologies have been developed to describe a document's semantic structure and annotate it with a semantic label. Some of them were designed for academic articles and others deal with other type of documents.

Ciccarese et al. developed an Ontology of Rhetorical Blocks (ORB) [5] to capture the coarse-grained rhetorical structure of a scientific article. ORB can be used to add semantics to a new article and to annotate an existing article. It divides a scientific article into three components: header, body and tail. The header captures meta-information about the article, such as it's title, authors, affiliations, publishing venue, and abstract. The body adopts the IMRAD structure from [24] and contains introduction, methods, results, and discussion. The tail provides additional meta-information about the paper, such as acknowledgments and references.

Peroni et al. introduced the Semantic Publishing and Referencing (SPAR) Ontologies [18] to create comprehensive machine-readable RDF meta-data for

the entire set of characteristics of a document from semantic publishing. It is used to describe different components of books and journal articles, such as citations and bibliographic records. It has eight ontologies to cover all of the components for the creation of RDF meta-data (DoCO, FaBiO, CiTO, PRO, PSO, C4O, BiRO and PWO). DoCO, the document components ontology [23, 6], provides a general-purpose structured vocabulary of document elements to describe both structural and rhetorical document components in RDF. This ontology can be used to annotate and retrieve document components of an academic article based on its structure and content. Examples of DoCO classes are chapter, list, preface, table and figure. DoCO also inherits another two ontologies: Discourse Elements Ontology (Deo) [7] and Document Structural Patterns Ontology [3].

Shotton et al. developed the Deo ontology to study different corpora of scientific literature on different topics and publishers. It presents structured vocabulary for rhetorical elements within an academic document. The major classes of Deo are introduction, background, motivation, model, related work, methods, results, conclusion, and acknowledgements. This ontology is very intriguing and relevant to our semantic annotation.

Monti et al. developed a system to reconstruct an electronic medical document with semantic annotation [17]. They divided the process into three steps. In the first, they classified documents in one of the categories specified in the Consolidated CDA (C-CDA) standard [8], using PDFBox [28] to extract text from CDA standard medical documents. Later, they split the document into paragraphs using the typographical features available in the PDF file. Finally, they identified key concepts from the document and mapped them to the most appropriate medical ontology. However, the paper lacks technical detail and an analysis of the results.

A Contextual Long short-term memory (CLSTM) [12] was used by Ghosh et al. for sentence topic prediction [10]. Lopyrev et al. trained an encoder-decoder RNN with LSTM for generating news headlines using the texts of news articles from the Gigaword dataset [15]. Srivastava et al. introduced a type of Deep Boltzmann Machine (DBM) for extracting distributed semantic representations from a large unstructured collection of documents [25]. They used the Over-Replicated $Softmax$ model for document retrieval and classification.

Tuarob et al. described an algorithm to automatically build a semantic hierarchical structure of sections for a scholarly paper [26]. They defined a section as the pair of the section header and its textual content. They employed a rule-based approach to recognize sections from scholarly articles and applied a simple set of heuristics that built a hierarchy of sections from the extracted section headers.

Most of the ontologies mentioned above are developed either manually or do not provide any technical details. Our ontology is an enhancement of Deo[7], developed using deep learning and embedding vector clustering to choose classes and the properties based on more than 1 million academic articles and a few hundred thousand business documents. Our approach is able to capture semantic meaning of different functional parts of a document by semantic annotation and semantic concept extraction, as described in our earlier research [19].

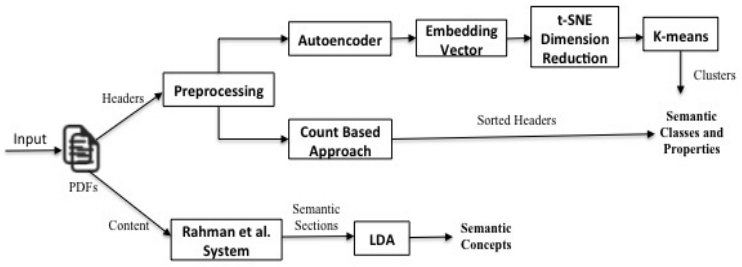



Fig. 1: Overall work flow of our system

## 3 Data Type and Document Category

In this research, we focus on extracting information from PDF documents. The motivation for focusing on PDF documents is the popularity and portability of PDF over different types of devices and operating systems. But automatic post-processing of a PDF document is not an easy task, since the objective of PDF rendering tools is not to support post-processing, but rather better visualization of the content. The rendering tools allow numerous equally valid ways of producing the same visual result and therefore no structure can reliably be derived from how the text operators are used. For experimental purposes, we choose PDF documents from academic articles and business documents, such as *arXiv* and RFP domains. We use a dataset which contains $1,121,363$ *arXiv* articles during or before 2016 released by Rahman et al.[20].

## 4 System Architecture and Technical Approach

This section describes the overall work flow of our system and the approach used for each of its parts.

### 4.1 System Architecture

The pipeline of our system is shown in Figure 1. The top-level, subsection and sub-subsection headers are retrieved from all the *arXiv* articles released by Rahman et al. [20]. After preprocessing, the headers are passed into Autoencoder. The embedding vector is dumped from the Autoencoder, which is passed through a t-SNE [16] dimensionality reduction. A k-mean [11] clustering algorithm is then applied on the reduced embedding vector. After preprocessing, a count based approach is also applied to retrieve all of the unique headers based on count. We also use our previous system [19] to get semantic sections, which are passed through LDA topic models to get domain specific semantic terms or concepts for each of the individual sections.

### 4.2 Technical Approach

In order to design a document ontology, we created a list of classes and properties by following the count-based and cluster-based approaches. In the count-based

Table 1: Classes for Ontology

| Document Type | Classes/Concepts |
|---|---|
| **Academic Article** | Introduction, Conclusion, Discussion, References, Acknowledgments, Results, Abstract, Appendix, Related Work, Experiments, Methodology, Proof of Theorem, Evaluation, Future Work, Datasets, Contribution, Background, Implementation, Approach, Preliminary |
| **RFP** | Introduction, Requirement, General Information, Conclusion, Statement of Work, Contract Administration, Appendix, Background, Deliverable, Contract Clauses |

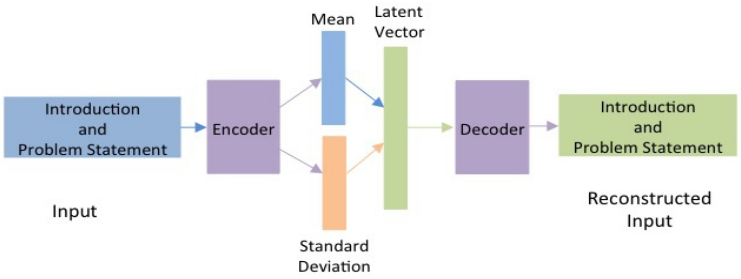



Fig. 2: Variational Autoencoder for Ontology Class Selection

approach, we first took all section headers, including *top-level*, *subsection* and *sub-subsection* which are basically headers from the table of contents of all *arXiv* articles. Then we removed numbers and dots from the beginning of each header, and generated the frequency for each header and sorted them. Based on a frequency threshold, we considered the section headers, which might be a class or concept for our ontology.

For the cluster based approach, we generated all section headers from the table of contents of all *arXiv* articles and developed a Variational Autoencoder and Convolutional Autoencoder to represent each of the section headers in a sentence level embedding, which is termed "header embedding" in our system. We applied Autoencoder to learn the header embedding in an unsupervised fashion, in order to produce good quality clusters. We then dumped the embedding vector from the bottleneck layer. Since this vector has high dimensionality and clustering high-dimensioned data often does not work well, we applied the t-SNE dimensionality reduction technique to reduce the dimensions of the embedding vector to just two dimensions. After dimensionality reduction, we used k-means clustering on the embedding vector to cluster the header embedding into semantically meaningful groups. We analyzed all clusters and all section headers from the count-based approach and came up with the classes or general purpose semantic concepts to design our document ontology.

We also applied a similar approach for section headers from RFP documents. To understand the sections of an RFP, we read [1] and discussed with experts from RedShred [2]. Table 1 shows classes from *arXiv* articles and RFPs, which were used to design a simple document ontology. The detailed descriptions are given below.

**Variational Autoencoder** A variational autoencoder is a type of autoencoder that learns latent variable models [9] for the input data. Instead of learning

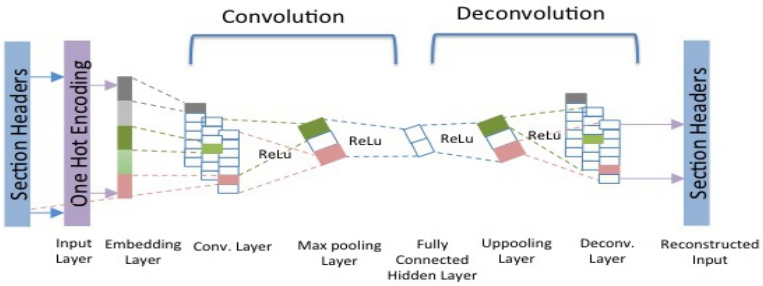



Fig. 3: Convolutional Autoencoder for Ontology Class Selection

an arbitrary function, the autoencoder learns the parameters of a probability distribution of the input data. The encoder turns the input data into two parameters in a latent space, which are noted as $\bar{z}$ and $z \log \sigma$. Then, randomly, a similar data point, $z$ is selected from the latent normal distribution using Equation 1.

$$z = \bar{z} + e^{z \log \sigma} * \epsilon \tag{1}$$

A final decoder maps these latent space points back to the original input data. The architecture of our VAE is given in Figure 2.

**Convolutional Autoencoder** A convolutional autoencoder is an autoencoder that employs a convolutional network to learn the parameters in an unsupervised way. Since our input is text, we use a Conv1D layer for both convolutional and deconvolutional parts of the network. The input text is converted into a *one-hot* encoding, which is passed into the embedding layer. Before encoding, we have Conv1D and MaxPooling1D layers with a *ReLu* activation function. The decoder starts with the deconvolution followed by an UpSampling1D layer. At the end, the decoder reproduces the original text. The architecture of the CAE is given in Figure 3.

**Document Ontology** After getting the classes from an analysis of the count- and cluster-based approaches, we designed an ontology for our input documents. The classes represent general purpose semantic concepts in our ontology. We also analyzed cluster visualization to get properties and relations among classes. Detailed results are included in section 5. Figure 4 shows our simple document ontology.

The document ontology includes classes that describe concepts induced from both *arXiv* academic articles and RFP documents. The top level "Document" class has two subclasses: "Academic Article" and "RFP". The Document class has "Category" that describes the type of document, such as Computer Science, Mathematics, Social Science, Networking, Biomedical and Software articles/RFPs. Both Academic articles and RFPs have contents, which are sections. These sections are the classes of different semantic concepts in a document. Both Academic articles and RFPs share some concepts, such as "Introduction", "Conclusion" and "Background". They also have their own concepts. For example, "Approach" and "Results" are available in Academic Articles whereas RFP has

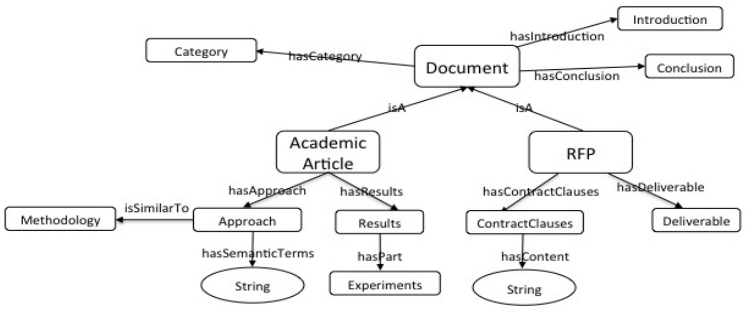



Fig. 4: The upper level of our document ontology, with rectangles representing classes and ovals properties. Additional classes are mentioned in Table 1.

"ContractClauses" and "Deliverable" concepts/classes. Due to space constraint, the classes are shown in Table 1.

Each of the classes/concepts has two properties "SemanticTerms" and "Content" which are represented by the relationships "hasSemanticTerms" and "hasContent". The data types for these two properties are String. The "hasSemanticTerms" property captures semantic topics applying Latent Dirichlet Allocation (LDA) to each section. Some concepts may have part, which is represented by a relationship "hasPart". For example, a concept "Results" has another subconcept "Experiments". Some of the concepts may be similar to another concepts, which is shown by a relationship "isSimilarTo". For example "Approach" and "Methodology" are two similar concepts.

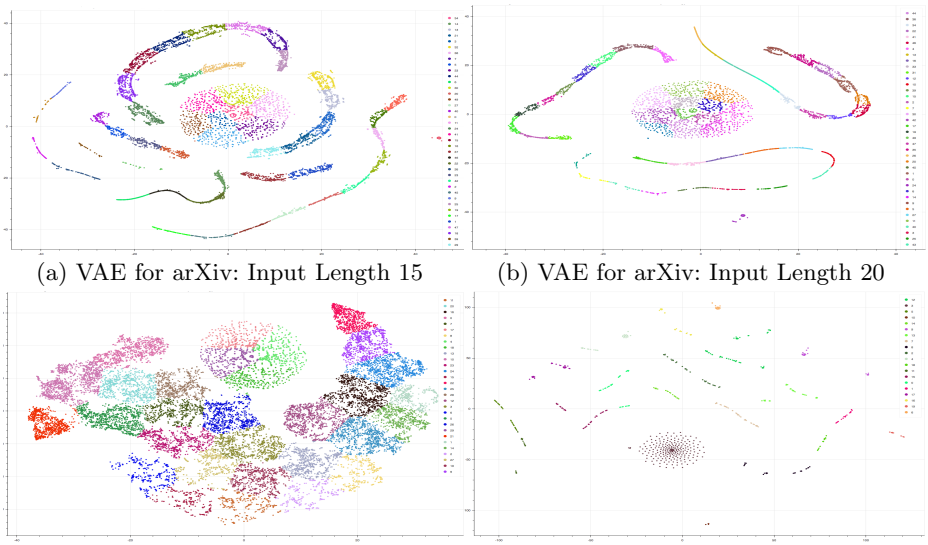
**Semantic Concepts using LDA** We used latent Dirichlet allocation (LDA) [4] to find domain-specific semantic concepts from a section. LDA is a generative topic model that is used to understand the hidden structure of a collection of documents. In an LDA model, each document has a mixture of various topics with a probability distribution. Again, each topic is a distribution of words. Using Gensim [22, 21], we trained an LDA topic model on a set of semantically divided sections. The model is used to predict the topics for any text section. A few terms that have the highest probability values of the predicted topics, are used as domain specific semantic concepts, or terms, for a given section. These semantic concepts are also used as property values in the document ontology.

## 5 Experiments and Results

In this section, we discuss the experimental setup followed by the detailed procedures. We also describe the results and the findings of each experiment and illustrate the results using comparative analysis.

### 5.1 Dataset

Using the dataset released by Rahman and Finin [20], we retrieved section headers from table of contents of all `arXiv` articles and applied some heuristics to remove unwanted text from the headers (e.g., numbers and dots) and downcased the text. The total number of unique section headers in our collection was



(a) VAE for arXiv: Input Length 15 (b) VAE for arXiv: Input Length 20

(c) CAE for arXiv: Input Length 15 (d) VAE for RFP

Fig. 5: t-SNE Visualization of Embedding Matrix Clusters

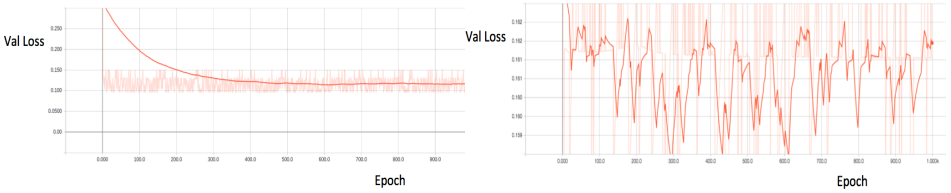
3,364,668 for all categories of *arXiv* articles. We used these section headers to get classes or concepts for ontology design as explained in section 4. We also retrieved section headers from only Computer Science articles. After applying a similar approach, we found 666,877 unique section headers from Computer Science articles. The experiments and results for all categories, as well as Computer Science, are described below.

### 5.2 Experiments on the arXiv Dataset

As described in section 4, we trained a VAE model to learn the header embedding for ontology design. We clustered the header embedding matrix into semantically meaningful groups and identified different classes for ontology. The VAE was trained with different configurations and hyperparameters to achieve the best results. We experimented with different input lengths, such as 10, 15 and 20 word length section headers. All section headers were converted into a multi-level *one-hot* vector.

We used 100 embedding dimensions, 100 hidden layers and 1.0 $\epsilon$ to learn latent variables. The *one-hot* vector was the input to the network, which was followed by an embedding layer with *ReLu* activation function. Then we had a dense layer to capture input features in a latent space. The model parameters were trained using two loss functions, which were a reconstruction loss to force the decoded output to match with the initial inputs, and a KL divergence between the learned latent and prior distributions. The decoder was used with a *sigmoid* activation function and the model was compiled with an *rmsprop* optimizer and KL divergence loss function.

We also trained a CAE model as described earlier and experimented with different hyperparameters. The model was trained with a *sigmoid* activation function, *binary_crossentropy* loss function, and *adam* optimizer. We also dumped




(a) For VAE (b) For CAE

Fig. 6: Validation loss

and clustered the embedding matrix to get semantically meaningful header groups.

**Results and Evaluation** Both the VAE and CAE models were trained in an unsupervised way to capture the semantic meaning of each section header. The outputs of the bottleneck layer were dumped and clustered after *t-SNE* dimensionality reduction. Figure 5a shows the visualization of *k-means* clustering with $k = 50$ and $inputlength = 15$ for VAE embedding after *t-SNE* dimensionality reduction. Similar visualization with $inputlength = 20$ is shown in Figure 5b. After analyzing both the visualizations, we observed that VAE models learned very well and were able to capture similar section headers together. We noticed that semantically similar section headers were plotted nearby. We also realized that semantically similar section headers were constructed gradually from one concept to another. For example, we detected a pattern in the graph where a sequence of concepts from "methods" gradually moved to "data construction", "results", "discussion", "remarks" and "conclusion". From this analysis, we could infer that VAE learned concepts over section headers in a semantic pattern. From the analysis and visualization of VAE, we found that the VAE models were capable of learning a manifold in the section header embeddings. This manifold can be used for computing semantically similar concepts in our ontology.

Figure 5c shows the clustering visualization of the embedding matrix generated by the CAE. After analyzing the cluster visualization, we observed that CAE models were not as good as VAE models. We also noticed that CAE models were not capable of learning a manifold in our dataset.

We achieved the best validation loss of 0.0947 for the VAE model and 0.1574 for the CAE model. Figures 6a and 6b show the validation losses over number of epochs for the VAE and CAE models, respectively. After analyzing these losses, we observed that VAE loss was steady after 600 epochs but that the CAE loss was oscillating after 20 epochs. This suggests that the VAE model performs better than the CAE model for our dataset. Since we achieved a better performance for VAE architecture on our dataset, we also trained VAE models for section headers from Computer Science articles, achieving performance metrics similar to those for all *arXiv* articles.

### 5.3 Experiment on RFP Dataset

We leveraged our existing collaboration with RedShred [2] to get a wide range of RFPs for our experiments. We trained VAE models for section headers from RFP documents. Due to a fewer number of section headers collected from RFP

Table 2: Comparative analysis of LDA models for semantic concepts

| arXiv Category | Word based LDA | Bigram based LDA | Phrase based LDA |
|---|---|---|---|
| **Mathematics - Algebraic Topology, Mathematics - Combinatorics** | algebra, lie, maps, element and metric | half plane, complex plane, real axis, rational functions and unit disk | recent, paper is, theoretical, framework, and developed |
| **Nuclear Theory** | phase, spin, magnetic, particle and momentum | form factor, matrix elements, heavy ion, transverse momentum and u'energy loss | scattering, quark, momentum, neutron move and gcd |
| **Computer Science - Computer Vision and Pattern Recognition** | network, performance, error, channel and average | neural networks, machine learning, loss function, training data and deep learning | learning, deep, layers, image and machine learning |
| **Mathematical Physics** | quantum, entropy, asymptotic, boundary and classical | dx dx, initial data, unique solution, positive constant and uniformly bounded | stochastic, the process of, convergence rate, diffusion rate and walk |
| **Astrophysics - Solar and Stellar Astrophysics** | stars, emission, gas, stellar and velocity | active region, flux rope, magnetic reconnection, model set and solar cycle | magnetic ray, the magnetic, plasma, shock and rays |

documents, we obtained different patterns, where most of the section headers were scattered all over the embedding space. Figure 5d shows the embedding visualization for RFP documents. It is interesting to notice that the VAE models for the RFP dataset are also capable of learning manifold, which can be used for calculating similar concepts.

## 6 Domain Specific Semantic Concepts using LDA

As described in section 4, we trained an LDA model using the divided sections generated from *arXiv* articles by the system developed in our earlier work [19]. The total number of training and test sections for the LDA were $128,505$ and $11,633$, respectively. We applied different experimental approaches using word, phrase and bigram dictionaries. The word-based dictionary contains only unigram terms whereas the bigram dictionary has only bigram terms. The phrase-based dictionary contains combination of unigram, bigram and trigram terms. All three dictionaries were developed from the training dataset by ignoring terms that appeared in less than 20 sections or in more than 10% of the sections of the whole training dataset. The final dictionary size, after filtering, was $100,000$. Different LDA models were trained based on various number of topics and passes. We ran the trained model to identify a topic for any section, which was used to retrieve top terms with the highest probability. The terms with the highest probability were used as a domain specific semantic concepts for a section.

For the evaluation, we loaded the trained LDA models and generated domain specific semantic concepts from 100 *arXiv* abstracts, where we knew the categories of the articles. We analyzed their categories and semantic terms. We noticed a very interesting correlation between the *arXiv* category and the semantic terms from LDA topic models, finding that most of the top semantic terms were strongly co-related to their original *arXiv* categories. A comparative analysis is shown in Table 2. After analysis of the results, we noticed that a bigram LDA model was more meaningful than either of the other two models.

## 7 Conclusion

Semantic annotation can be described as a technique of enhancing a document with automatic annotations that provide a human-understandable way to represent a document's meaning. It also describes the document in such a way that

the document is understandable to a machine. Using our developed ontology, we built a system to annotate a PDF document with human understandable semantic concepts from the ontology. The system, along with the research components and ontology, will be available soon[1]. In this research, We have presented Variational and Convolutional Autoencoders which capture general purpose semantic structure and different LDA models for domain specific semantic concept extraction from low level representation of large documents. Our approaches are able to detect semantic concepts and properties from a large number of academic and business documents in an unsupervised way.

## Acknowledgment

The work was partially supported by National Science Foundation grant 1549697 and a gifts from IBM and Northrop Grumman.

## References

1. Sections of an rfp (2017), `http://www.wipp.org/resource/resmgr/gm5_podcasts_rev/RFP_Help.pdf`, accessed 22-October-2017
2. Redshred (2018), `https://www.redshred.com/`, accessed 26-January-2018
3. Angelo Di Iorio, Fabio Vitali, S.P.: Document structural patterns ontology (2017), `http://www.sparontologies.net/ontologies/pattern/source.html`, accessed 09-October-2017
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
5. Ciccarese, P., Groza, T.: Ontology of rhetorical blocks (orb). editors draft, 5 june 2011. World Wide Web Consortium. http://www. w3. org/2001/sw/hcls/notes/orb/(last visited March 12, 2012) (2011)
6. Constantin, A., Peroni, S., Pettifer, S., Shotton, D., Vitali, F.: The document components ontology (doco). Semantic Web **7**(2), 167–181 (2016)
7. David Shotton, S.P.: Discourse elements ontology(deo) (2017), `http://www.sparontologies.net/ontologies/deo/source.html`, accessed 09-October-2017
8. Dolin, R.H., Garber, L., Solutions, I.: Hl7 implementation guide for cda® release 2: Consolidated cda templates for clinical notes (us realm) draft standard for trial use release 2
9. Eisenstein, J., O'Connor, B., Smith, N.A., Xing, E.P.: A latent variable model for geographic lexical variation. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. pp. 1277–1287. Association for Computational Linguistics (2010)
10. Ghosh, S., Vinyals, O., Strope, B., Roy, S., Dean, T., Heck, L.: Contextual lstm (clstm) models for large scale nlp tasks. arXiv preprint arXiv:1602.06291 (2016)
11. Hartigan, J.A., Wong, M.A.: Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) **28**(1), 100–108 (1979)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
13. Holden, D., Saito, J., Komura, T., Joyce, T.: Learning motion manifolds with convolutional autoencoders. In: SIGGRAPH Asia 2015 Technical Briefs. p. 18. ACM (2015)

[1] These will be available in summer 2018.

14. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
15. Lopyrev, K.: Generating news headlines with recurrent neural networks. arXiv preprint arXiv:1512.01712 (2015)
16. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(Nov), 2579–2605 (2008)
17. Monti, D., Morisio, M.: Semantic annotation of medical documents in cda context. In: International Conference on Information Technology in Bio-and Medical Informatics. pp. 163–172. Springer (2016)
18. Peroni, S.: The semantic publishing and referencing ontologies. In: Semantic Web Technologies and Legal Scholarly Publishing, pp. 121–193. Springer (2014)
19. Rahman, M.M., Finin, T.: Deep understanding of a documents structure. In: 4th IEEE/ACM Int. Conf. on Big Data Computing, Applications and Technologies (December 2017)
20. Rahman, M.M., Finin, T.: Understanding the logical and semantic structure of large documents. arXiv preprint arXiv:1709.00770 (2017)
21. Rehurek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (2010)
22. Rehurek, R., Sojka, P.: Gensim–python framework for vector space modelling. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic (2011)
23. Shotton, D., Peroni, S.: Doco, the document components ontology (2011)
24. Sollaci, L.B., Pereira, M.G.: The introduction, methods, results, and discussion (imrad) structure: a fifty-year survey. Journal of the medical library association **92**(3), 364 (2004)
25. Srivastava, N., Salakhutdinov, R.R., Hinton, G.E.: Modeling documents with deep boltzmann machines. arXiv preprint arXiv:1309.6865 (2013)
26. Tuarob, S., Mitra, P., Giles, C.L.: A hybrid approach to discover semantic hierarchical sections in scholarly documents. In: Document Analysis and Recognition (ICDAR), 2015 13th International Conference on. pp. 1081–1085. IEEE (2015)
27. Turchenko, V., Chalmers, E., Luczak, A.: A deep convolutional auto-encoder with pooling-unpooling layers in caffe. arXiv preprint arXiv:1701.04949 (2017)
28. Wikipedia: Apache pdfbox — wikipedia, the free encyclopedia (2016), `https://en.wikipedia.org/w/index.php?title=Apache_PDFBox&oldid=740366251`, accessed 20-September-2016