Context-Aware Data Loss Prevention for Cloud Storage Services

Yuya Jeremy Ong*, Mu Qiao[†‡], Ramani Routray[†] and Roger Raphael[§]
College of Engineering, Penn State University, PA, USA*
IBM Almaden Research Center[†], San Jose, CA, USA
IBM Analytics Group[§], San Jose, CA, USA
Email: yjo5006@psu.edu, {mqiao, routrayr, rogerr}@us.ibm.com

***Abstract*—With the wide adoption of hybrid cloud, there are many potential risks that need to be mitigated to ensure that the utilizations of services are at their optimal levels. One of the major risks that has garnered much attention is maintaining maximum security and confidentiality for sensitive information. Detecting sensitive content at near real-time and at cloud scale has become a critical first step for organizations to prevent data loss and comply with data privacy laws and regulations. Proactive detection raises security awareness at the early stage and thus can be used to govern how the information should be managed, protected, and utilized in the hybrid cloud. In contrast to traditional dictionary or policy-based approaches, we introduce a system that detects sensitive content by leveraging its semantic contextual information through various machine learning and deep learning techniques at different levels of granularity within the document, and is the first of its kind.**

***Keywords*-context-aware, sensitivity, data loss prevention, hybrid cloud, and deep learning**

I. Introduction

Data loss prevention (DLP) has become a significant security strategy in the era of cloud computing. DLP software products or services protect sensitive information so that they will not be accidentally or maliciously shared outside of the defined trust zones. For example, many enterprises are using cloud storage for online sharing and collaboration, utilizing services such as IBM Softlayer, Box, and Google Drive [1, 2, 3]. Prior to uploading data from on-premises to shared directories in the cloud, it is critical to ensure that all the sensitive information are properly identified, protected, or redacted.

With the recent development in cloud technologies, many enterprises have shifted towards the adoption of hybrid cloud, which utilizes a mix of on-premises, private cloud and public cloud. This is driven by a paradigm shift in the way digital services are handled and managed through the use of different cloud infrastructures. According to a recent study, 85% of enterprises report that hybrid cloud is accelerating the digital transformation in their organizations [4]. Given the multi-tenant and heterogeneous infrastructure of hybrid cloud, enterprises have become aware of the potential vulnerabilities in sensitive information being leaked or stolen by an adversary or competitor. For instance, the majority of private clouds are compliant under the HIPAA regulations (Health Insurance Portability and Accountability Act). However, the transmission of personal health data from the private cloud to a public cloud would increase such risks and vulnerabilities, and should not be permitted. Not having full auditing capabilities for their own data increases such risk significantly. Therefore, it is critical that enterprises have a DLP strategy with an instituted process to ensure the security of sensitive data and the regulatory compliance in the hybrid cloud ecosystem.

[‡]Corresponding Author

The transmission and dispersion of files in the hybrid cloud increase the likelihood that adversaries obtain sensitive information due to the increased redundancy of data. One of the best prevention policies is to contain the spread of files as soon as they are identified to have sensitive information. This requires the capability of detecting these sensitive information in real time. Formally, we define sensitive information as a piece of data owned by entities or individuals, which, if potentially lost, damaged or compromised, bares significant financial and integrity damages to the enterprise or person of interest. There are many different forms of sensitive information, such as personal identifiable information (PII), sensitive personal information (SPI), personal health information (PHI), confidential business information, and other domain-specific sensitive information.

The current and more prevalent approach to detecting sensitive information is dictionary or rule based, which essentially matches the given text against a list of predefined regular expressions and keywords. For example, the regular expression describing nine digits in the format of "123-45-6789" can be used to detect social security number. Similarly, a set of keywords, such as "driver's license", "credit card number", and "nationality", can also be included in the list. This approach depends on a very well defined and maintained dictionary. Because it only performs a direct string matching, without considering the semantic context of the words, the detection precision can be very low. In addition, the possible combination of data formats, context, word root variations, abbreviations can be on such a massive scale that a pattern matching approach is rendered ineffective or impractical. All it takes is one breach on one critical piece of data that can produce far-reaching impact in its loss for the person or company concerned. One of the biggest challenges is being able to detect sensitive content in context, which can drive our understanding of the risk exposures. For example, the word ”Asian” is sensitive personal information regarding ethnicity/race in the sentence “Tony Chen immigrated to U.S. from an Asian country to start a business at age 23”, but not sensitive in ”Every May, the company hosts an Asian American heritage celebration month”. In another example, in the sentence “Company ABC will revise the technical solution document and pricing documents and resubmit them to Company XYZ”, both “ABC” and “XYZ” are sensitive company names since the context is related with business engagement and contract. On the other hand, “ABC” and “XYZ” are not sensitive information in the sentence “Company ABC and Company XYZ today announced a business agreement to deliver a highly secure, first-of-its-kind cloud service that uses private networks rather than the public Internet”, which is in the context of the press release.

In this paper, we introduce a context-aware DLP system, which provides real-time sensitivity detection at various hierarchical degrees of granularity: from the document, to sentence, down to token level. The document level detection is scalable to a large corpus of text documents within the hybrid cloud. It can quickly identify sensitive documents on a high level. The sentence level detection allows users to have a further deep understanding of data sensitivity on each sentence. Finally, the token level detection identifies all the sensitive entities within the document. This hierarchical detection gives users the flexibility to identify sensitive information at different levels of granularity based on their specific needs. For example, the user may not only need to know which business document is sensitive, but also need to redact or sanitize client information from such document for privacy concerns. Different from all previous works, our system detects the sensitivity at different levels by leveraging contextual semantic information through deep learning methods. Specifically, to classify if documents are sensitive or not, our system learns the context by representing them as dense vectors through document embeddings. To determine the sensitivity of sentences, it learns the context through a convolutional neural network. Finally, our system detects sensitive tokens by considering the specific context they reside through a recurrent neural network. In contrast to traditional rule-based and bag-of-words (BOW) based detection approaches, our context-aware detection methods have shown superior performance.

We have deployed the system in a hybrid cloud architecture inside a large organization, which has both on-premise storage and cloud storage service utilizing the Box enterprise network. Our deep learning based detection models inspect the files in real time, allowing the system to immediately respond to any behavior or events violating data sensitivity compliance. The system also provides an interactive dashboard for hybrid cloud auditors and administrators to gain insights into the sensitive data distribution and trend, as well as monitor user behaviors within the Box enterprise network. By providing the visualization and monitoring functions to the auditors, they can take proactive and immediate action when necessary. For example, the system can provide alerts when a user suddenly starts to upload many sensitive documents within a very short time or when certain users begin to transfer sensitive files out of the trust zone.

The rest of the paper is organized as follows. We review the related work in Section II. The overall system architecture is introduced in Section III. Our context-aware sensitivity detection methods are discussed in Section IV. We present the experimental results and discuss lessons learned in Section V and VI. Finally, we conclude in Section VII.

II. Related Work

There are two general schools of work related with sensitive information detection: rule-based and model-based methods. Many DLP systems, such as SPIRION, CUSpider, and CipherCloud [5, 6, 7], utilize regular expressions, custom keywords, or domain specific entity corpora like ICD numbers, PCI codes, and national drug codes, to detect sensitive information. These systems require the use of a massive dictionary and the detection precision is low due to false positives. Through the recent advances in natural language processing, the understanding of text in semantic context has been significantly improved through machine learning models. As opposed to traditional rule-based systems, these model-based systems can learn from user data and enforce dynamic policies in a significantly more effective manner.

The model-based detection methods can be further divided into the categories of unsupervised and supervised learning approaches. For example, Sanchez et. al. [8] developed an unsupervised information-theoretic approach to assessing the degree of sensitiveness of terms according to the amount of information they provide. A corpus-based association rule mining method is applied to detect sensitive inferences from text documents [9]. Supervised machine learning methods, such as Naive Bayes and Support Vector Machines (SVM), have been applied to detect privacy leaks from tweets using a carefully crafted training data [10, 11]. Cumby and Ghani [12] developed a semi-automated system to redact private information from text documents by framing it as a multi-class classification problem. Symantec introduced a Vector Machine Learning as part of its data leakage software, which detects sensitive information from unstructured data by training a statistical model on both sensitive and non-sensitive samples [13]. The use of model-based methods provides several key advantages over rule based methods. Notably, two main advantages come from (1) the lower computational cost and less disk space for maintaining dictionaries and indexes, and (2) the ability to adapt to the dynamically changing environment by training models on new data.

To the best of our knowledge, no model-based methods have leveraged the semantic context in detecting sensitive information. In addition, most existing detection systems are either only focused on the document level or token level. Our context-aware system enables real time sensitivity detection at different levels of granularity through advanced deep learning methods, and is the first of its kind.

III. System Architecture

In this section, we will introduce the architecture of the developed context-aware DLP system. The technical components of our system are implemented as microservices within a hybrid cloud, which allow for both horizontal and vertical scaling based on the requirements of the enterprise, as well as easy integration with different APIs, such as those provided by IBM's Bluemix for document parsing. Figure 1 shows the overall system architecture. By utilizing the microservice architecture as our development model, the components in Figure 1 can be implemented in different programming language or framework, thus making it language agnostic which significantly reduces the integration effort with other cloud platforms. For example, different cloud storage services or platform services on the Software as a Service (SaaS) layer can be easily integrated with our context-aware DLP microservice through the REST API based communication protocol.

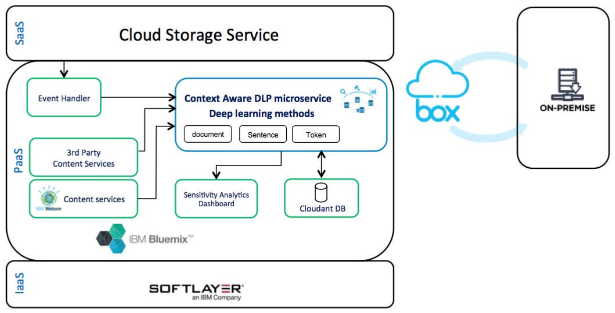

Figure 1: System architecture of the developed context-aware DLP system

Our system consists of three main technical components: (a) a filesystem event handler, (b) a context-aware DLP microservice, and (c) sensitivity analytics detection system. In the following sections, we will briefly describe the functionality of each microservice.

A. Event Handler

The event handler acts as a bridge between the cloud storage service and the DLP microservice, which detects any file/object create, read, update or delete (CRUD) operations within the cloud. Detection of asynchronous CRUD operations are handled by a webhook based reporting function. The cloud storage service on the SaaS layer sends a REST based API call to the DLP microservice's entry point to enable the sensitivity detection service. By allowing such asynchronous reporting of CRUD operations, the system is able to inspect files in real time, and immediately respond to any anomalous behavior that violates data sensitivity regulations. Furthermore, it can also perform on-demand sensitivity detections specified by the cloud auditor or administrator. For example, the detection can be enabled on a collection of files, where the same entry point of the DLP microservice is used to invoke a detection process for each file.

B. Context-Aware DLP microservice

We have developed several context-aware methods to detect the sensitive information at different levels of granularity through several advanced deep learning models. These models are wrapped in a microservice that analyzes the documents in real time through the use of GPU clusters, and returns detection results on document, sentence, or token levels based on the auditor's request. One of the key advantages to utilize the microservice architecture is the easy swap of detection models for different levels of sensitivity analysis through an integrated infrastructure which can handle and manage the modeling process of the DLP microservices.

C. Sensitivity Analytics

Our system also includes a sensitivity analytics dashboard for auditors to monitor and gain insights into the sensitive data distribution and trend, as well as monitor user behaviors in the cloud. Figure 2 shows a temporal heat map, which is used to monitor the change in sensitivity index, a measure quantifying the amount of sensitive information contained within specified cloud space. By providing these visualization and analytical functions to the auditors, they can immediately respond to activities that may violate security compliance. The provided analytics dashboard can not only help to inform auditors, but also can be used as a tool to predict and prevent potential risks from occurring before they cause actual harm to the organization.

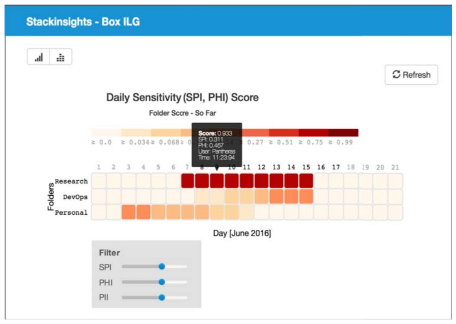

Figure 2: Temporal sensitivity heat map of a cloud space

IV. Context-aware Sensitivity Detection Models

To better contain the risk of documents from being compromised against threats, our system needs to proactively detect sensitive information with high precision while not sacrificing the overall runtime performance. Formally, we define our problem as having a set of entities (documents, sentences, tokens) E and a model M to determine for all $e \in E$, whether the given entity e is "sensitive" or "non-sensitive". In the following subsections, we describe the specific classification objective and the corresponding model architectures that we have developed to detect the sensitivity of content at different hierarchical levels.

A. Document Level

We have a set of documents, where the objective of our model is to classify each document as being "sensitive" or "non-sensitive". The output of the model can also be interpreted in a probabilistic manner indicating the chance the document being uploaded onto the cloud is potentially a sensitive document. In contrast to traditional Bag-of-Words (BOW) model, we represent each document by a dense vector which is trained through a deep neural network. The trained vectors can better represent the semantics in documents because they are learned through the prediction tasks considering various context of the words. As a comparison, we also represent each document by a topical vector which is learned from the Latent Dirichlet Allocation (LDA) model. We have experimented with several different types of feature representational methods to encode our document as a single feature vector, which is then used to train a classifier model such as a Logistic Regression model or Support Vector Machines. In the following, we will briefly describe each model.

1) Bag-of-Words: The Bag-of-Words (BOW), a common and simple feature representation in natural language processing, is utilized as a baseline model in our work. We utilize BOW in document sensitivity classification by representing each document as a bag of tokenized words, tracking only the word frequency but not taking into account their order. The feature vectors are then used to train a logistic regression classifier.

2) Latent Dirichlet Allocation: Latent Dirichlet Allocation (LDA) is a generative probabilistic model which can find latent semantic topics within a collection of documents. Those discovered topics are attributed to a specific distribution of words with the assumption that the mixture of topics inside a document has a Dirichlet prior. Hence, each document can be characterized by having a mixture of topical distributions and thus provides insights into the semantics the documents contain. However, exact model inference in LDA is generally intractable. Approximated methods, such as Gibbs sampling, can be used to estimate the model parameters. We refer interested readers to [14]. Finally, given the formulated implicit document-topic distribution, we can then construct a SVM classifier model using a RBF kernel to classify the documents based on the derived topical feature vectors.

3) Document Embeddings: Feed-forward neural networks have been used to create embeddings which retain the semantics of words in a myriad of natural language processing tasks [15]. Notably, Mikolov et. al [16] have developed a scalable architecture with the Continuous Bag of Words (CBOW) model and the Skip-Gram model, to train word embeddings for generating semantic preserving feature representations of unstructured data using negative sampling. In extrapolation to the work of Mikolov et. al [16], the architecture is extended to accommodate variable-length sequences of texts including sentences and documents [17]. In this work, we have utilized this skip-gram variant of the feature embedding modeling as it is shown to produce higher quality results with regards to understanding semantic relationships of words.

Specifically, we train a document embedding neural network on a given set of documents to derive the corresponding embedded feature vector for each document. A logistic regression model is then trained on these feature vectors. For an unseen document, we utilize an inferential approach to deriving the approximate embedding values and then obtain the feature vector. The trained model is then applied to predict whether that document is sensitive or not.

B. Sentence Level

In the sentence level detection, we treat each sentence (or a line of text, delimited by a new line - depending on the structure of the document) as a single atomic unit of evaluation, and determine if the content within that sentence is "sensitive" or "non-sensitive". We apply different deep learning methods to detect the sensitivity due to the level of granularity. Specifically, we evaluate two methods: (1) Long Short Term Memory Networks (LSTM), and (2) Convolutional Neural Networks, which are able to capture the context of words in the sentence. We use the same Bag-of-Words (BOW) method described in the previous section as our baseline model.

1) LSTM Networks: Recurrent neural networks (RNN), which accept data with temporal or sequential dependencies, have been applied to tag word sequences or predict the next word given the input of a sequence of words that appear before the target prediction [18]. In our work, the RNN will output the sensitivity label given the input of a sequence of words in the sentence.

However, the parameter inference process to train RNN, commonly through methods such as backpropagation through time (BPTT), is often very difficult due to issues caused by the vanishing or exploding gradient, where the magnitude of the gradient can increase or decrease exponentially within the beginning or end of the sequence of the network [19]. This issue makes training RNN especially challenging, especially when the output prediction is conditioned on long distance features. To mitigate this issue, the Long Short Term Memory recurrent neural network is proposed to utilize memory based gates to help remedy the vanishing or exploding gradient problem.

In our LSTM architecture, we first obtain the word embedding feature vector for each token in the sentence. For every sentence in the training data, its corresponding word embedding feature vectors are then provided as input to the LSTM network, which contains a single LSTM layer followed by a sigmoid output. As illustrated in Figure 3, we implement the many-to-one mapping of words to the corresponding sensitivity label. The input word embedding vector for each individual word in the sentence will be mapped to the output of the network at the very end of the sentence, denoted by an "end of line" or EOL token. By passing the input through each LSTM cell and then predicting the corresponding sensitivity label by a sigmoid function at the final output, we can determine whether the given sentence or string of text is sensitive or not, indicated by either "1" or "0" respectively.

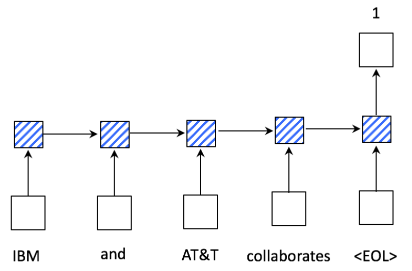

Figure 3: Sentence level LSTM architecture

2) Convolutional Neural Networks: Convolutional neural networks (CNN) have been used in many domains, such as computer vision, image processing, and speech recognition. Notably CNN are well suited for their applications in learning low level features and producing good representations of the raw data [20]. They have shown robust performance when applied to distributed or discrete word embeddings, without considering the semantic or syntactic structure of a language [21].

We apply a 1D convolutional neural network to determine the sensitivity of a sentence, which is a linear list of tokens. The main component is a 1D temporal convolution module. We also utilize a 1D version of the max-pooling layer which is commonly used in computer vision. It has been empirically shown that the max-pooling operation helps to improve the performance by allowing deeper layers to be trained [21]. The non-linearity utilized in the network is a rectified linear unit (ReLU). In our architecture, the word embeddings for tokens in the sentence are provided as input to the 1D convolutional neural network. The filters in the convolutional layer imply contextual windows around the word. After applying the convolution operation, the vectors go through a 1D max pooling operation and a dense fully connected neural network layer. The resulting values are finally passed into a sigmoid function where the sensitivity of the sentence is determined. The model utilizes an Adam based optimizer for its gradient optimization method.

C. Token Level

In this section we introduce two models which classify a specific token from a sentence as being sensitive or non-sensitive. We utilize two different variants of the Long Short Term Memory (LSTM) Networks including (1) a standard LSTM architecture, and (2) a bidirectional LSTM Network.

1) LSTM Networks: To detect token sensitivity, we modify the previous many-to-one LSTM architecture in the sentence level detection to a many-to-many architecture. An exemplar LSTM network is shown in Figure 4, where "IBM" and "AT&T" are sensitive company names, labeled as "1"s, while the remaining non-sensitive tokens are labeled as "0"s in the output sigmoid layer. It also uses the word embedding to represent each token in the document corpus. The word embedding feature vectors will then go through the LSTM layer and output the weights to sigmoid functions where the sensitivity of the token is determined. As opposed to determining the output of the network on the EOL token in sentence level detection, we instead evaluate the output from each LSTM cell to determine the token sensitivity based on its current context. We also use an Adam based optimizer in the network.

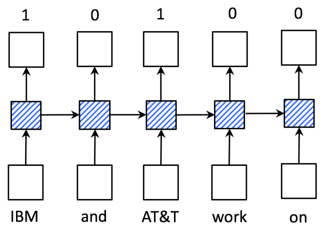

Figure 4: Token level LSTM architecture

2) Bidirectional LSTM Networks: To better utilize the context of a token, we leverage the temporal dependencies among tokens in both directions (forward and backward) through a bidirectional LSTM network [22]. We use both the past features (via forward states) and future features (via backward states) for a specific sequence window. The bidirectional LSTM network is illustrated in Figure 5. As we can see, the contextual information of a word can be leveraged through the learning in the hidden layer which consists of both forward and backward passes. Therefore, the sensitivity classification of a token does not only depend on the features of that token, but is also conditioned on other words present within the context. In our implementation, we have a similar setup as the previous LSTM architecture, except utilizing a bidirectional layer for both the forward and backward passes.

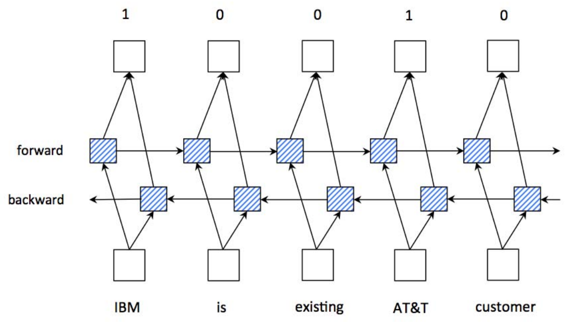

Figure 5: Token level bidirectional LSTM architecture

V. Experiments

In this section, we evaluate the performance of the proposed context-aware methods when they are applied to detect document sensitivity at different levels of granularity. We will introduce the real world dataset utilized in the experiments, the training process, parameters, and the final model performance. All the experiments are conducted using 10-fold cross validation.

A. Dataset

Most works in this area dealing with sensitive content and materials do not have datasets that can be easily shared or distributed among researchers and institutions. Therefore, we have curated our own dataset which is comprised of both sensitive and non-sensitive documents within a large organization, such as business engagement contracts, customer facing materials, technical solution documents, and public newsletters. We build a simulated collection of documents which can exist within the cloud of a typical enterprise. Our curated base dataset is comprised of 530 documents consisting of common file types such as Microsoft Word, Excel, Powerpoint, and PDF files. Approximately half of the documents (268) in the collection are marked as sensitive. In each of our experiments, we utilize this dataset and alter the annotation for each type of model we are building and preprocess the documents accordingly. All training and test samples, prior to being processed in the model, undergo text extraction, tokenization, removal of punctuation and stop words, and normalization (converting to lowercase).

B. Document Level

In our document level experiment, we have trained three different models as described in Section IV. The input to each model is a collection of raw text documents which are preprocessed by the previously described steps. For the document embedding, we obtain feature vectors which are of 300 dimensions, trained from a feed-forward neural network through 15 epochs. As for the LDA model, the topic size is empirically set to 5. Each document is therefore represented as a topical feature vector of five dimensions. A Bayesian-based hyperparameter optimization routine is applied to find the best hyperparameters. Across each of the models, we perform a 10-fold cross validation and verify the average Area Under the Curve (AUC) metric. In our classification problem, the positive class is "sensitive" while the negative class is "non-sensitive". Specifically, the precision is defined as $tp/(tp + fp)$, where tp is the number of true positives and fp is the number of false positives. The recall is $tp/(tp + fn)$, where tp is the number of true positives and fn is the number of false negatives. Accuracy is defined as the ratio between the number of samples that are correctly classified and the total number of samples. The F1 score is computed as $2 \times (precision \times recall)/(precision + recall)$, which is a weighted average of the precision and recall. The performance of each method is presented in Table I, followed by the AUC curve shown in Figure 6. From the experimental results, we find that the document embedding model works the best in terms of all metrics. On the other hand, LDA model performs the worst, which may be caused by the small number of topics.

Table I: Document level models

Model	Accuracy	Precision	Recall	F1
Bag-of-Words	0.82	0.72	0.80	0.76
LDA	0.55	0.66	0.58	0.62
Doc2Vec	**0.83**	**0.83**	**0.83**	**0.82**

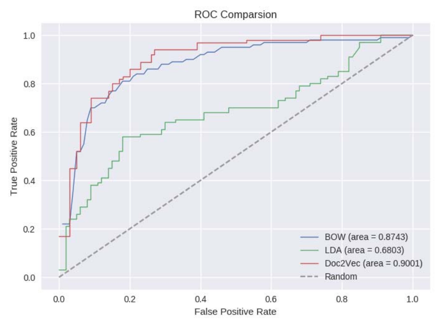

Figure 6: Document level AUC

C. Sentence Level

In our sentence level experiment, we have trained three different models: BOW, LSTM, and CNN. For this experiment, we manually extract sentences from the documents and label them. To make sure there is sufficient contextual information in each sentence, we only consider sentences which have at least five tokens. We prepare a dataset with

Table II: Sentence level models

Model	Accuracy	Precision	Recall	F1
Bag-of-Words	**0.91**	0.89	0.89	0.89
LSTM	0.84	0.89	0.87	0.88
CNN	**0.91**	**0.92**	**0.95**	**0.93**

474 sentences, among which 316 are labeled as sensitive and the rest as non-sensitive. The size of word embedding in both LSTM and CNN is set to be 128. The word embedding feature vectors are provided as input to the LSTM network with 64 output dimensions. In CNN, the 1D convolutional layer utilizes 250 filters, which are of length 3 and use the ReLU activation function. After passing the convolutional layer, the vectors go through a 1-D max pooling and a dense fully connected neural network layer which is of 250 output dimensions. We run 5 epochs for training both LSTM and CNN. The performance of each model is reported in Table II. Across each of the models, we have generated an average AUC value as shown in Figure 7. From the results, we find that CNN outperforms the other methods in terms of both precision and recall.

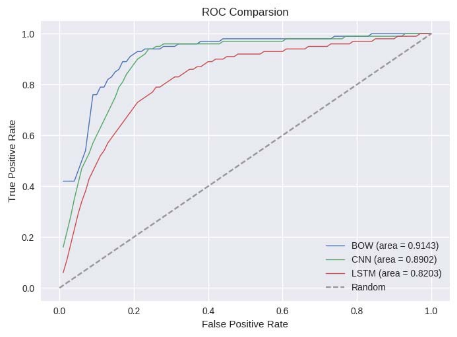

Figure 7: Sentence level AUC

D. Token Level

In the token level experiment, we have trained two models: the standard many-to-many LSTM and Bidirectional LSTM (Bi-LSTM). For a baseline comparison, we have also included a traditional dictionary based approach, which classifies a token as sensitive if it matches a keyword or regular expression in the dictionary. For this experiment, we have modified our dataset by annotating any occurrence of the company names, such as “IBM” or “AT&T”, as sensitive, if they originate from a sentence labeled as sensitive in Section V-C. All the remaining tokens are marked as non-sensitive. Our dataset contains a total of 9,427 tokens, with only 390 marked as sensitive. The size of word embedding in both LSTM and Bi-LSTM is set to be 128. The embedding feature vectors are then provided as input to the LSTM and Bi-LSTM networks with 64 output dimensions. We use the

Table III: Token level Models

Model	Accuracy	Precision	Recall	F1
Dictionary	0.97	0.91	0.48	0.63
LSTM	**0.99**	0.92	**0.97**	0.94
Bi-LSTM	**0.99**	**0.94**	**0.97**	**0.96**

Table IV: Models running time

Model	**Training (sec)**	**Inference (ms)**
Document - Doc2Vec	187.60	4.00
Sentence - CNN	7.77	36.40
Token - Bi-LISTM	329.704	71.42

same setting as the LSTM in sentence level detection. Due to the highly imbalanced ratio between sensitive and non-sensitive tokens in the training data, we run a much larger number of epochs (i.e., 100) to train both models in order to achieve the desired accuracy. As shown in Table III, both LSTM and Bi-LSTM perform significantly better than the dictionary based approach. The dictionary based approach achieves a very high overall accuracy as it classifies a large number of non-sensitive samples correctly. By utilizing the contextual information of target tokens through both forward and backward passes, Bi-LSTM has shown superior overall performance among all the methods.

E. Running time analysis

All the deep learning models (Doc2Vec, CNN, LSTM, and Bi-LSTM) are trained on a single Nvidia 1080 Ti GPU using Keras (Theano as the backend), a python based library. The remaining models are trained on a Intel Core i7-3820 processor with 16 GB of RAM. Table IV shows the average running time of the models for training and inference (classifying a single sample) over 10-fold cross validation. Due to limited space, we only show the running time of the model with the best performance at each detection level. As we can see, the inference time for each model is less than 100 milliseconds, therefore, allowing for a near real-time detection in practice.

VI. Lessons Learned

In this section we discuss some of the lessons learned during our system implementation, as well as some of the key observations on utilizing model based methods for detecting sensitive information.

One of the major caveats of using model based methods is the interoperability of the model, that is, being able to work with various types of documents regardless of its overall distribution of topics and document structures. Since every organization and user is different, the attributes of such documents can vary across different domains and industries. This can ultimately yield different training features and thus affect the detection performance. To improve the model robustness, more work in improving the overall feature representations of text data should be considered. Especially cases where the collection of documents are spread across multiple domains can potentially jeopardize the robustness of the models. One way to improve the robustness of the models is to utilize an ensemble-based approach, such as bagging or boosting techniques across the models we have introduced in this paper to improve the overall precision and recall of the classification.

Another area to consider when integrating model based detection into the cloud infrastructure is online training. Instead of training the model from scratch, we can learn the parameters of new models based on existing weights and inference. The use of online learning can be especially beneficial for scaling towards handling data with high volume and keeping up with the high velocity of incoming data. This can significantly reduce the overall computational overhead required to maintain the model's integrity, and can be potentially faster than a re-indexing process utilized in a dictionary based approach. However, further studies should be conducted to analyze how such training methods can affect the overall performance of the model. Furthermore, to maintain the quality and integrity of the models, a model evaluation and training process handler should be implemented to (a) constantly test the performance of the model trained on existing data, (b) determine an appropriate policy to evaluate whether the model can suffice with online learning or would require a complete retraining of the model, and (c) devise a scalable cloud-based infrastructure for handling these necessary modeling tasks.

VII. Conclusion

In this paper, we have introduced several model based methods for detecting sensitive content in the hybrid cloud by leveraging the contextual semantic information through advanced deep learning models. The developed data loss prevention system can detect sensitive content at various degrees of granularity within the document, and immediately respond to any behavior or events violating data sensitivity compliance. We demonstrate that through the effective use of model based methods, the system is capable of mitigating future threats that can help Chief Information Officers and Chief Data Officers in retaining strong security integrity of their hybrid cloud by taking preventative measures.

References

[1] Softlayer. http://www.softlayer.com/cloud-storage [Online]

[2] Box. http://www.box.com/ [Online]

[3] Google Drive. https://www.google.com/drive/ [Online]

[4] "Growing up hybrid: Accelerating digital transformation", IBM Center for Applied Insights Study, InterConnect 2016.

[5] SPIRION. https://www.spirion.com/ [Online]

[6] CUSpider. https://www.columbia.edu/acis/security/spider/ [Online]

[7] CipherCloud. https://www.ciphercloud.com/ [Online]

[8] D. Sanchez, M. Batet, and A. Viejo, "Detecting sensitive information from textual documents: an information-theoretic approach", in Proceedings of International Conference on Modeling Decisions for Artificial Intelligence, Vol. 7647, pp. 173-184, 2012.

[9] R. Chow, P. Golle and J. Staddon, "Detecting Privacy Leaks using Corpus-based Association Rules", in Proceedings of the ACM SIGKDD, pp. 893-901, 2008.

[10] A. Caliskan-Islam, J. Walsh and R. Greenstadt, "Privacy Detective: Detecting Private Information and Collective Privacy Behavior in a Large Social Network," in Proceedings of the 13th ACM Workshop on Privacy in the Electronic Society (WPES'14), 2014.

[11] H. Mao, X. Shuai, and A. Kapadia, "Loose Tweets: An Analysis of Privacy Leaks on Twitter", in Proceedings of the 10th ACM workshop on Privacy in the electronic society (WPES'11), 2011.

[12] C. Cumby and R. Ghani, "A machine learning based system for semiautomatically redacting documents", in Proceedings of the 23rd conference on Innovative Applications of Artificial Intelligence, pp. 1628-1635, 2011.

[13] Symantec. https://support.symantec.com/en_US/article.TECH219962.html [Online]

[14] D. Blei, A. Ng, and M. Jordan"Latent Dirichlet Allocation", Journal of Machine Learning Research vol. 3, 993-1022, 2003.

[15] G. Mesnil, T. Mikolov, M. Ranzato, and Y. Bengio. "Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews". arXiv:1412.5335, 2014.

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality", in Proceedings of NIPS, pp. 3111-3119, 2013.

[17] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents", in Proceedings of ICML, pp. 1188-1196, 2014.

[18] J. L. Elman, "Finding structure in time", in Cognitive Science, 1990.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory", in Neural Computation, 9(8): 1735-1780, 1997.

[20] R. Johnson and T. Zhang, "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks", in Proceedings of NAACL HLT, 2015

[21] X. Zhang, J. Zhao, and Y. LeCun, "Character-level Convolutional Networks for Text Classification", in Proceedings of NIPS, pp. 649-657, 2015.

[22] Z. Huang, W. Xu and K. Yu, "Bidirectional LSTM-CRF Models for Sequence Tagging", arXiv:1508.01991, 2015.