CSI6900- Graduate Research Project – Winter 2019

Harish Rajapuram: 300035296

Master of Computer Science - hraja101@uottawa.ca

CSI6900- Graduate Research Project Report submitted

To

Professor Dr. Diana Inkpen

University of Ottawa

School of Electrical Engineering and Computer Science

800 King Edward Avenue, Room 5015

Ottawa, Ontario, Canada, K1N 6N5

diana@site.uottawa.ca

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Automatic Text Classification of NATO documents with Recurrent Neural Networks (RNN)

Harish Rajapuram – 300035296

Under Supervision of Prof. Dr. Diana Inkpen

ABSTRACT

Deep learning models in Natural Language Processing (NLP) have improved the state-of-the-art in Text Categorization, Speech Recognition, Machine Translation [4], and many other subfields. With advancement in the digital world, the increasingly available documents in the web created a spur in the researcher's community to organize them in various categories for ease of use and document retrieval. Central to this idea of text classification, traditional classifiers built as linear models using the machine learning algorithms have suffered from data sparsity, noise, dimensionality and poor convergence in generalization tasks. To overcome the above-mentioned issues, we will experiment and build various deep learning models by using recurrent neural network structures to learn the semantic information of the text documents as a supervised learning. Subsequently, we will evaluate the performance of these deep learning models with the traditional machine learning models for comparison. In our experimentation, we have compared the performance of our RNN and other variants with the baseline models of SVMs.

1. INTRODUCTION

Document modeling or Text classification is one of the extensive research areas which will be helpful in applications such as text categorization, spam detection, information retrieval, sentiment analysis, and document summarization. It is essential to understand that traditional text classification methods use a bag of words model to classify the text documents which may end-up being suffered by the curse of dimensionality [2]. In addition to that, these traditional methods often ignore the word ordering and actual contextual information of the words. Furthermore, they may produce unsuitable features space in capturing the semantics of the words. To give an example the word "fair" can be used in both ways to describe the appearance and the other as a reasonable quantifier. Hence, the idea of deep representation of sentences by sequence models raised hope among researchers. The sequence models using the word level embeddings can keep track of its distributed representation in a sentence to assuage the data sparsity problem.

The referred sequence models often termed as Deep Neural network (DNN) models, which have been remarkable in achieving higher accuracy in document modeling or classification. Convolutional Neural Networks (CNN) and Recurrent neural networks (RNN) are two mainstream architectures for document modeling or Text classification. In general, Text classification comprises presenting the sentences as meaningful features and applying machine learning algorithms for the classification task. Usually, the learning of word representation needs more computations due to the larger number of parameters involved in the learning process. Representing words as a vector and feeding the collection of words as a sequence to neural network embedding layer is widely followed initial step in text classification task in deep learning. The words in a given dataset can be learned using the other datasets references by linking the relationship between our dataset with other datasets [3]. These learning can be transferred to the embedding layer of our model to capture the semantic information in a sentence. As the embeddings are already pre-trained, usage of these features gives the model a push in the learning process.

Owing to the capability of capturing word sequences of any length and long-term dependencies, RNN has been phenomenal in learning semantics of the given input sentences. Since the cells in RNN can propagate the historical information via a chain like structure for every word in sequence and produce output depending on previous computation. Hence, over the training, these cells have the capability of capturing the contextual information. Although, RNN suffer from later words of sequences becoming more dominant than the previous sequences, may cause an inability in capturing "long-term dependencies". However, careful modeling of the RNN's with adding memory units or callback features while training and additional layers of attention networks to capture the sentence level dependency can alleviate the vanishing gradient problems [4-5].

In this paper, we have implemented various deep learning methods of RNN which has the capability in sharing semantic information of word sequences among the hidden layers. Also, these models will be tested on the North Atlantic Treaty Organization (NATO) documents and performance improvements will be compared with respect to baseline model Support-vector machine (SVM). Furthermore, these RNN configurations will be optimized on parameter learning and dropout inferences in LSTM and GRU models in document classification for accuracy improvements [5]. The short memory units known as cells in LSTM and GRU are like block-boxes which considers the current input x_t, hidden state h_{t-1} constitute together on regulating the input states [11]. Internally these cells will decide what to keep in and what to erase from memory to capture long-term dependencies. Based on the anecdote of the existing architecture, our experimentation results favored LSTM and GRU architectures over the traditional RNN's. In the next sections, we refer the terms document classification and text classification interchangeably.

2. RELATED WORK

Distributed representation of sentences and learning process in the neural network may need little extra effort when compared with the linear classifiers, owing to that cost, it improves the accuracy in text classification task [1]. Since deep representations hold a promising case of considering prior representations at each time step in classification. A linear machine learning models like Random Forest, SVM heavily dependent on n-gram, TF-IDF features, that consists of each word token in isolation for the classification task. This may lead to a situation of erroneous capture or understanding of sentence transitions, in case of words representing negative sense to some point and suddenly spins towards positivity at the end. However, in Neural networks, they have the capability to see the words like "though", "besides", "exiting", "in contrast" etc. in contextual space. For example, given the sentence "We had expected nothing but a tranquil nature. On the contrary, it was dirty and pale", presents spin-off in its entire representation, where sentence implied something on expectation and concluding with negative along with the expectation.

Joachims in [6] has introduced a Transductive SVM, which implements a hyperplane by understanding the statistical properties of the text for classification. Certainly, the method introduced a scalability factor to handle many documents and features by introducing a tranductive plane over regular inductive approach in minimizing the misclassification errors. However, there are a lot of open questions pointed in the research regarding the suitable text representation, margin alignments and effects of combined indictive feature space for performance improvements. Furthermore, in information retrieval systems and text classification, researches have introduced various SVM approaches [7] like linear SVM, kernel SVM, Hierarchical SVM, and ensemble pipeline SVM, etc. to improvise the overall performance.

In contrast with the linear models, Deep learning models have been remarkable in handling high dimensional data and capture the long-term dependencies for a better understanding of the semantics of the given input. Authors in [8], has proposed an approach of "multi-task learning", where each task is assigned

to different layers which can be shared between RNN's whenever needed in the learning process. Their experiment results suggest that their model improves the system performance by sharing information and exploring common feature space. Lai et.al in [9] introduced the concept of recurrent convolutional neural networks and this approach was able to outperform regular RNN implementation. Here, the contextual information was obtained through the recurrent networks and convolutional neural networks are used to understand the text representation since the most important words will be selected in max pooling layer to actively reduce the feature space for the classification task.

Furthermore, the Hierarchical Document Classification in [10] has combined the multiple deep learning approaches to produce document classification at each level. Also, their experimentation shows the improvement of classification accuracy by using LSTM and GRU. In addition, there were ideas proposed by authors in [1] to combine LSTM and other CNN for classification task and the results are pretty much improvement over the simple LSTM or CNN architectures. Inspired by all of the implantation ideas, In our approach we will concentrate on RNN and its other variants of deep learning architectures for text classification. This report will start with the explanation of RNN and its related implementation and experimentation on NATO documents for classification.

3. DOMAIN DESCRIPTION

This report will elaborate on text classification comparison with linear SVM model and its pipeline approaches to the simple RNN to GRU and LSTM models. The RNN models will be stacked as sequential models and trained with respect to various parameters such as dropout and recurrent dropout to reduce the overfitting cases and results will be compared for evaluation.

3.1. Support Vector Machines (SVMs)

SVMs main objective in text classification scenario is to find a marginal hyperplane in N-dimensional space where N represents number of features which can be used to classify the data points. Hence, the data points which fall on either side of the hyperplane can be attributed to that class. As shown in figure 1, the hyperplane separating two different classes will be a line, data point which near to the hyperplane and uniquely identifies or represents the class can be defined as a support vector. This separation completely depends on the distribution of data and its dimensionality, as the dimensionality of the data increases visualization of support vectors will become complicated.

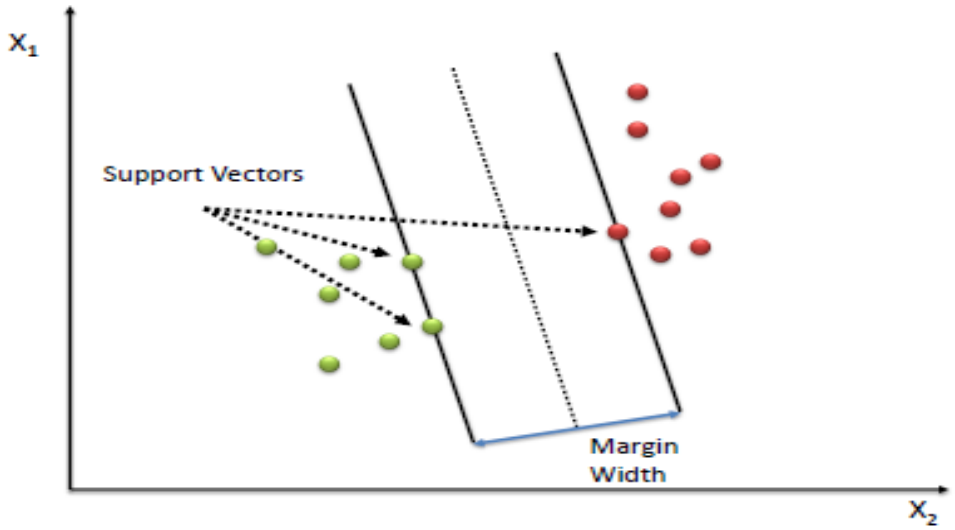

Fig. 1. Support vectors in SVM

In practice, text classification often involves the multiclass learning problem. This multi-class learning problem can be solved by pairwise comparison between the classes as one versus one style of comparison. This style of comparison will end up generating n(n-1) total pairs, where n is the number of classes in the comparison. Besides that, on-versus-all comparison considers one class vs all other class like a binary SVM to generate the hyperplane for classification.

3.2. Recurrent Neural Networks

Recurrent in RNN process the output of the current time step as an input to the next time step. At each time step, RNN has the capability in processing the arbitrary sequence of input length by recursively applying a transition function from previous time steps. The general formulation of RNN can be explained by using the equation 1 below, where h_t represents the state of the system aspects from past sequences of input at time t and x_t refers to the input arrived at time t [12].

$$h_t = f(h_{t-1}, x_t, \theta) \qquad (1)$$

With the addition of weight matrices and bias matrices while computation using the pretrained embeddings, equation 1 can be reformulated as shown in equation 2.

$$h_t = W_{rec}\, \sigma(h_{t-1}) + W_{embed}\, x_t + b \qquad (2)$$

Here the W_{rec} is the recurrent matrix weight, W_{embed} is embedded weights obtaining from the word embeddings for each word of sequence which are pretrained, b is bias and σ is "element-wise function". Figure 2 shows a simple recurrent neural network diagram with an output y. Traditionally, sequence modeling of RNN follows a procedure of mapping input variables to fixed sized vector of representation and feed forward to the next layers and subsequently to the SoftMax layer for the classification task. Here, irrespective of the sequence length, "the model will always have the same input size, since it is specified in the transition from one state to another state" [12]. As previously stated, the problem associated with traditional RNN is the later sequences becoming more dominant and vanishing gradient problem while backpropagation. To address these issues, researches have suggested more futuristic models like GRU and LSTM.

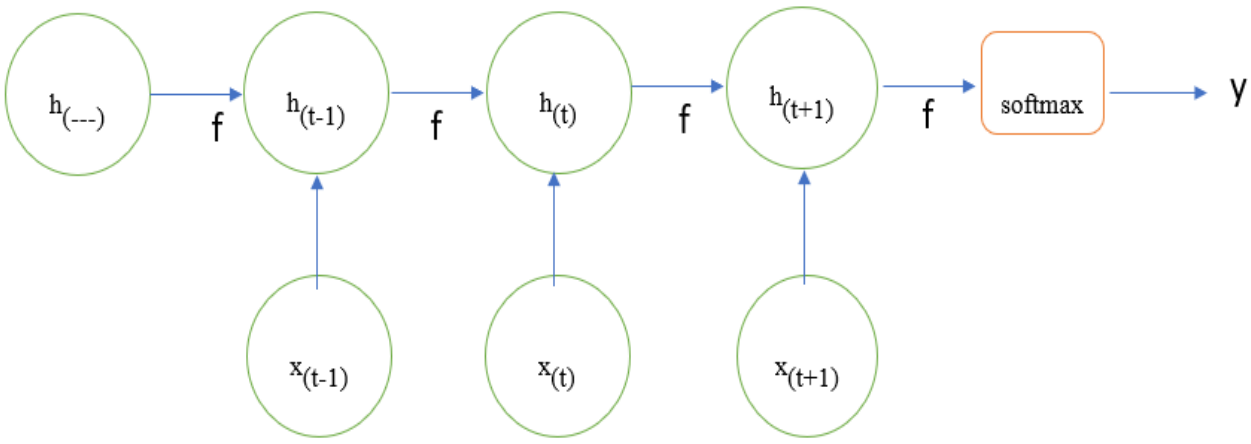

Fig. 2. Simple RNN unfolding architecture with recurring inputs and previous states for classification.

3.3. LSTM and GRU

"Long Short-Term Memory networks" – called LSTM [13], were introduced by Hochreiter & Schmidhuber in 1997 to alleviate the problem associated with learning long-term dependencies. These LSTM and Gated Recurrent Units (GRU) are nothing but RNNs, have an extra feature of having a separate memory cell inside it. Regular RNN structure has only recurrent structure of the network layer and the corresponding activation function to process the sequence vectors, whereas GRU operates through the reset gate and update gate. In the case of LSTM, the system uses input, forget and output gates. The input gate invokes how much of the cell state to be considered, forget gate computes how much of the existing learned sequence can be forgotten and finally output gate regulates the cell state parameters to next layers.

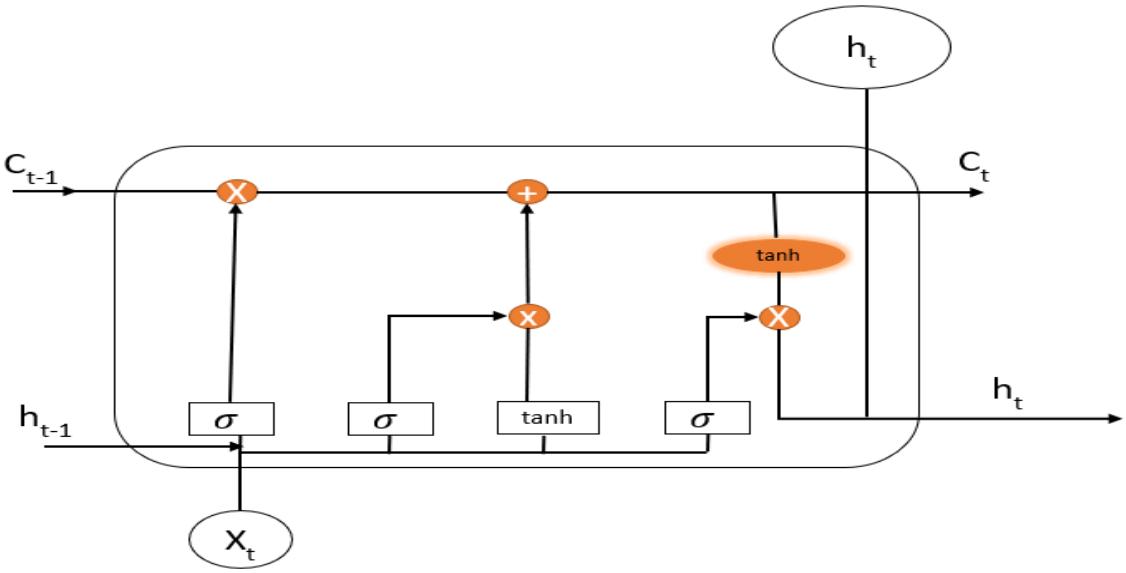

Fig. 3. LSTM cell structure

The main distinction between the LSTM and GRU is with respect to the overall exposure of information to the next layers. In LSTM, the overall cell state exposure is controlled through input and forget gates, however in GRU exposure of cell state is visible to other units in the network. From the above figure 3, which depicts the LSTM cell structure where h_{t-1} represents the previous output of hidden state, x_t refers to current input and c_t refers to current cell state at the current time step t. The equations for input, output and forget gates can be written as follows [1].

$$i_t = \sigma\,(W_i \cdot [h_{t-1}, x_t] + b_i) \qquad (3)$$

$$f_t = \sigma\,(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$q_t = \tanh\,(W_q \cdot [h_{t-1}, x_t] + b_q)$$

$$C_t = f_t \times c_{t-1} + i_t \times q_t$$

$$o_t = \sigma\,(Wo \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \times \tanh(c_t)$$

From the above equations, i_t denotes input gate, f_t denotes forget gate, o_t denotes output gate at time step t. Furthermore, σ is a sigmoid function, tanh is a hyperbolic function, both functions are used as an activation for that input cell. As previously mentioned, each gate performs the assigned task in assessing the sequential input and ensuring the information will not be lost in due process and regulating vanishing or exploding of gradients problem. Coming to GRU, it is a simplified mechanism of LSTM without any cell state C_t, hence its cell state is exposed to other units in the network.

4. EXPERIMENTS ON NATO DOCUMENTS

This section describes NATO documents, preprocessing and creating deep learning models using RNN, GRU and LSTM and training on the corpus for classification problem. In this text classification task, we consider the classes of interest are *Unclassified, Confidential, Secret and Restricted*. This multiclass text classification statement of interest prompted us to understand the real-world data preprocessing issues, labeling and creating machine learning models for text classification on North Atlantic Treaty Organization Documents. Furthermore, it has given an opportunity to dig deep into the optimization of deep learning models, in comparison to baseline models such as SVMs.

4.1. Data Preprocessing

Around 31000 NATO PDF documents were collected previously for text classification purpose. Having said that, our experimentation starts from the point of converting PDF to text documents and applying preprocessing steps. Previous researches have suggested the usage of pdftotext conversion or ghost script to extract the text content from the PDF's. However, we felt that the conversion process is not fully automated and need more inspection, hence we decided not to use these methods. Certainly, there were two other research paths to extract complete text without losing any information, one is using the python library pdfminer to extract or using "Optical Character Recognition (OCR)" [14]. As most of the documents (99%) are PDF's and to use OCR, we must convert them to jpeg's and use English and French OCR tesseract training sets. To the best our knowledge, the popular packages of OCR conversion in python are by using API's supplied in pytesseract or tesseract-OCR.

As previously stated, a major problem associated with OCR, they accept only images as the main input source and are not cost-efficient pertaining to our task. Besides that, OCR takes the extra effort of deciding the language of document over iteration and convert them to text. Based on all the observations we have opted pdfminer, which considers the pure Unicode interpretation of each byte hence captures all the text irrespective of the pdf document language. A Python script was built using the pdfminer library and iteratively ran around the PDF documents from converting it to text. With this approach, all the documents were successfully converted to text, but a few of them suffered from the noisy data inside it. The main reason could be the presence of handwritten documents in pdf format and few words/sentences were strikethroughs in the documents. For example, strikethrough at the word "overall" is converted to over-ûll, the script tried every effort to capture all the content including official signature with dates and stamps, by doing so it has generated some sort of noisy text data in between. Especially, in some part of the documents each character in a word was separated by single space (ex: A g e n c i e s c o n t a i n e d In the E n c l o s u r e s). We have tried every effort to clean and preprocess the documents for our text classification.

Labeling of documents was handled by generating a label corpus file by writing the top five and bottom five lines from each document. In fact, most of the documents within label corpus already have their class label, and it made our process easier to build a python script to generate final label file based on the

conditions we define in our script. We built a regular expression search case with positive and negative lookahead to capture the class labels (Unclassified, Confidential, Secret and Restricted) to extract the complete word and its surrounding characters. The reason behind this process is to eliminate similar word extensions to the existing class labels. For example, “secretariat and secretary” are the addition of chars to the “secret” class label. In addition to this, pdfminer has generated some erratic conversions for a few documents by clubbed neighboring characters to the word, such an example is “confidential1, secretnato” etc. As the collection contains both French and English, we have chosen all the English only documents for our training and testing with more than 5 lines. Also, it is safer to assume that there were a smaller number of “mislabeling” in our labeling process since some of the documents clubbed two different class text inside of it and we have ignored this kind of issues and diverted them to the first encountered class. As we have chosen the lines at the top, mid, and bottom for larger documents for our training from the corpus, we assumed this will not affect much to accuracy.

4.2 Feature Extraction and Embeddings

Feature extraction is the crucial step before conducting any of the machine learning algorithms on the dataset. Considering the premise of neural networks and its input, the words in text corpus must be represented as a meaningful representation of numbers as an input. For the baseline model such as SVM, we have used count-based term frequency-inverse document frequency (tf-idf) for feature extraction. Hence, in our approach, we will be using the count of N-gram as feature selectors. The vectorization of N-grams encompasses the capability of representing the surrounding words in the generalization task. Hence, we thus choose the tf-idf vectorization in our experiments with SVM, based on grid search parameter, we will finalize unigram or bigram representation.

Deep learning models using RNN and other variants, distributed representation of words will be suitable for better accurate models. According to researches in [15], the neural network can be converged to local minima with the unsupervised pre-trained procedure. In this work, we utilize the trained vector-space embedding from 200-dimensional Glove [16]. Unlike the word2ve, the Glove embedding learns based on the co-occurrence matrix that captures the meaning of each word in vector space.

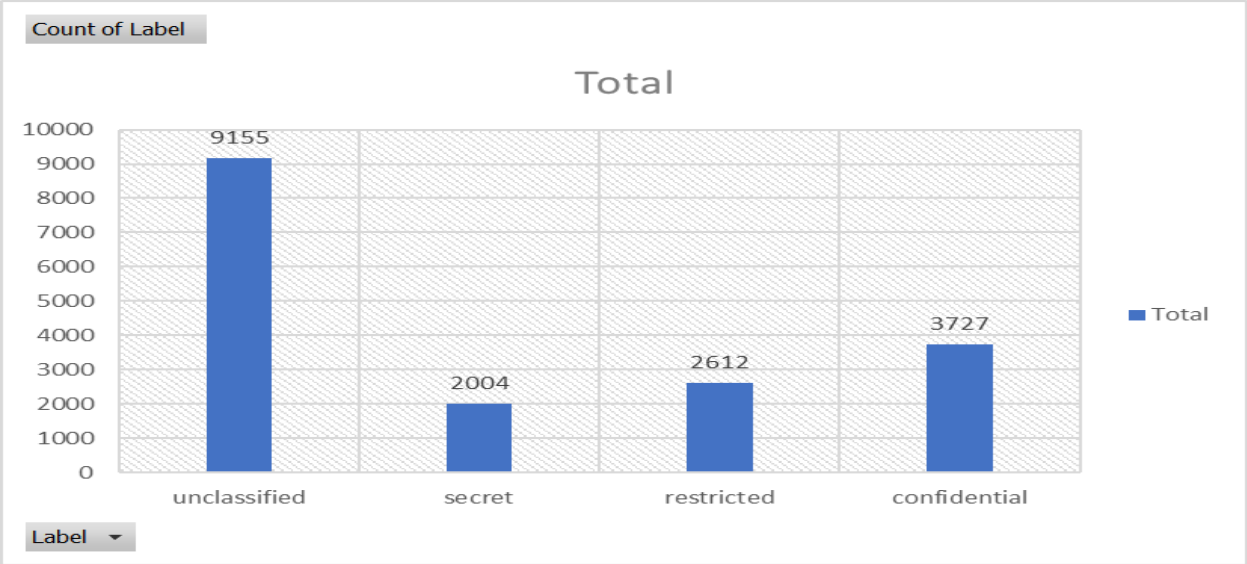

Fig. 4. Class distribution in NATO documents

Using the 200-dimensional Glove embedding, we will map the vector space of each word to the words in our corpus. In nutshell, the overall vocabulary size of the corpus is 273612, though we limit this huge number of unique tokens to 75000 and max sequence length to 500. With glove dimensions of 200, the input embedding layer will be the size of 500 × 200 and model will be trained on imbalanced class. Above figure 4 shows the total number of documents with the class distribution. In our experiments, we will use 70% of the documents for training and 10% for validation while training and 20% for testing. While training, we will be early stopping our model based on validation loss, to make sure the model will not overfit further. Next section explains about the models and results.

5. EXPERIMENT EVALUATION

In this section, we perform the empirical evaluation of the above mentioned RNN and other variant models. We start by experimenting on the multiclass corpus using the baseline model of SVM, we chosen two variants of SVMs as for performance comparison. One model of SVM is by building a simple linear classifier which supports scaling LinearSVC and other being the one vs rest classifier with a little tweak on class imbalance problem. On the other side, deep learning models used in the comparison are simple RNN, GRU and Bidirectional LSTM with attention [17]. All deep learning models use below definitions and parameter values for training the model.

RNN: batch size on each epoch 55, number of epochs 10, Total number of GRU and LSTM cells as 100 with two hidden layers, learning rate 0.001, both dropout and recurrent dropout with 0.2.

		Train Docs	Validation Docs	Test Docs
	Total Number of Documents -17498	12598	1400	3500
	Methods			**Test Accuracy**
	LinearSVC (SVM)	63.48%		***62.20%***
	LinearSVC + OneVsRest balanced	71.07%		***71.28%***
	Linear SVC with Grid Search	NA		***64.03%***
Baseline	***LinearSVC + OnVsRest balanced and Grid Search***	NA		***73.54%***
	Text RNN	75.28%	69.21%	***68.68%***
	Bidirectional LSTM with attention [17]	81.19%	73.43%	***71.40%***
Deep learning Models	***GRU classifier + dropout***	76.60%	72.36%	***73.85%***

Table 1. Baseline and Deep learning Models and corresponding accuracy

To extend the optimization ideas of deep neural networks, we have used recurrent dropout and RMSprop optimizer to reduce the validation loss and eventually to increase the accuracy. The RMSprop is almost like the gradient descent algorithm which has some set of functional similarities with Momentum. Momentum

reduces its oscillations in only one direction and converges to the global minimum of a cost function while learning the parameters. Whereas, RMSprop reduces the oscillations in vertical directions while converging to the global minimum, which allows the freedom to choose a higher magnitude of learning rates. Also, it has wide support in deeper neural networks involving RNN with a higher number of parameter learning. However, it may suffer from sparse gradient issue while learning the parameters, so one must consider these drawbacks while choosing RMSprop. Coming to the Dropout, it is a regularization technique widely used in researches community while building the deep neural networks. "The dropout in neural network prevents the co-adoption of hidden units, which may be formed by omitting the actual feature detectors of parameters". As confirmed by most of the researches [5], we have used the dropout only for non-recurrent units in the network.

Table 1 describes the baseline and deep learning models and their corresponding accuracies. Certainly, regular baseline model i.e. linear SVM performed the worst in all baseline categories with 62% accuracy. To scale well against the larger data, we have chosen the linear kernel in SVM's. In order to improve upon the obtained scores, we have performed grid search options, balancing the classes since there exists a class imbalance. Also, we have removed all stop words to increase the accuracy of SVM's. Certainly, the available options from scikit-learn pipeline objects, allowed us to look after the hyperparameters for our SVM models. The grid search selection has suggested the usage of bigram vectorization (range - (1,2)) as a feature for the classifier. Hence, from the above table, SVM classifier with the inclusion of suggested pipeline parameters and balancing of dataset gave the highest accuracy of 73.5%. Furthermore, the stop words were removed from the corpus while classifying the documents, which bolstered the model to perform well.

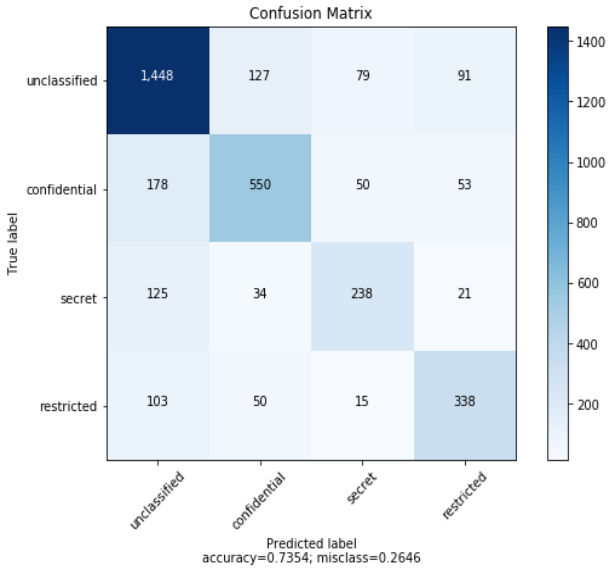

Fig 5. SVM - OneVsRest (confusion matrix)

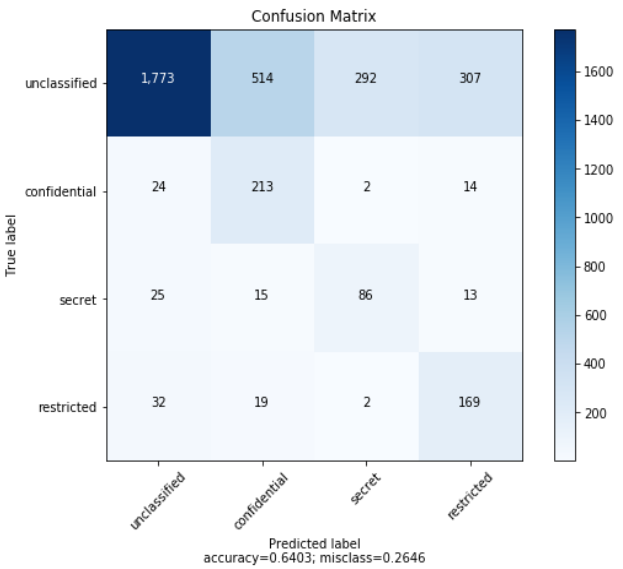

Fig 6. SVM (confusion matrix)

Figure 5 shows the confusion matrix of SVM on 3500 test documents, As the classifier was optimized and trained on balanced classes (OneVsRest), it was able to perform well with an accuracy of 73.5%. However, in Figure 6, the confusion matrix of optimized linear SVM classifier shows the poor results on test data. Though this model is optimized with Grid search parameters, class imbalance while testing has generated poor results and misclassified most of the documents to the majority class(unclassified).

A) Deep learning Models and Results

The results have shown that the deep learning models for NATO document classification task are performed better when compared to the baseline SVMs. As for as the RNN models are considered, the simple RNN model has performed worst with 68.7% accuracy. As a matter of fact, the simple RNN is nothing but a recurrent unit applied on maximum sequence length vectors at each layer and activations to move forward to the next layers. As pointed by the researches in [5] [8-10] the exploding gradient problem on larger sequences and holding memory for a longer time makes them vulnerable to performance degradation. Also, in this simple RNN architecture, we weren't considered any of the optimization criteria except RMSprop. Hence this model has performed worse in above mentioned deep learning RNN models.

Preface to the model training, the corpus has a lot of unique tokens, and this may lead to the learning process more cumbersome in deep learning, though we were considering a maximum number of unique tokens to 75000(vocabulary size) in our experiments. To give an example, some of the documents in our data has multiple pages, referenced as "page1, page2..." etc. occurrence of these words in a large number of documents concludes "page" as a frequent unique token. Hence, in some cases, our embeddings will consider these words leaving behind the less occurred unique words. Also, the conversion process (pdfminer) created some important word sequences like "timely response needed" to 1. "time ly response ne eded", 2. "tim e response need" which may be due to the inability in interpreting characters like "e", "o" etc. as documents are too old and characters were masked for some reason. Eventually, this makes the tokenizer to interpret each word as separate tokens and confuses the learning algorithm while predicting the class label. In addition to these, labeling of the documents have faced a lot of challenges especially on the class labels "secret and restricted". In some documents, even though the label infers confidential, but inside text expresses interest related to the secret document. Owing to these reasons the system has performed poor, including baseline model SVMs.

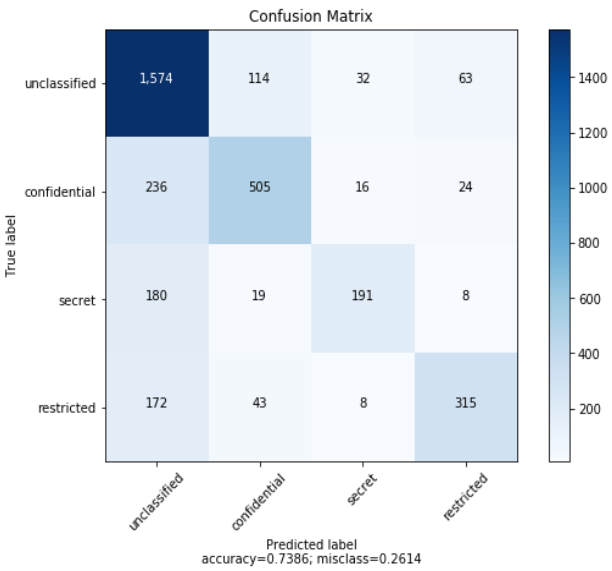

Fig 7. GRU classifier + dropout

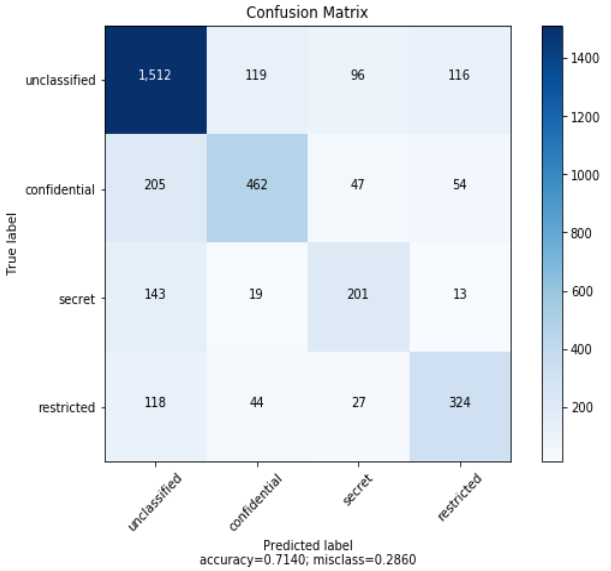

Fig 8. Bidirectional LSTM + Attention

Finally, other RNN variants of Bidirectional LSTM with attention performed superior on training and validation data, since the model has the capability in remembering the sequences at each iteration which includes both forward pass and backward pass. Bidirectional LSTM feeds the data from beginning to end and from end to beginning and learns larger sequences intact to train on the data and perform well on validation. Also, the inclusion of attention [17] layer merges word level features at each time step to the sentence level feature vector to capture the semantics of the sequences learning. However, based on the above reasons it has performed moderately on test data producing the accuracy of 71.40%. The final model of GRU with the inclusion of dropout and recurrent dropout layers to leave behind the impotent feature sequence vector mappings, generated less dense layers while learning eventually in generating the accurate model. In overall, this implementation methodology has performed equally well on training, validation and testing data. Above figure 7 and 8 shows the confusion matrices of GRU and Bidirectional LSTM classifiers.

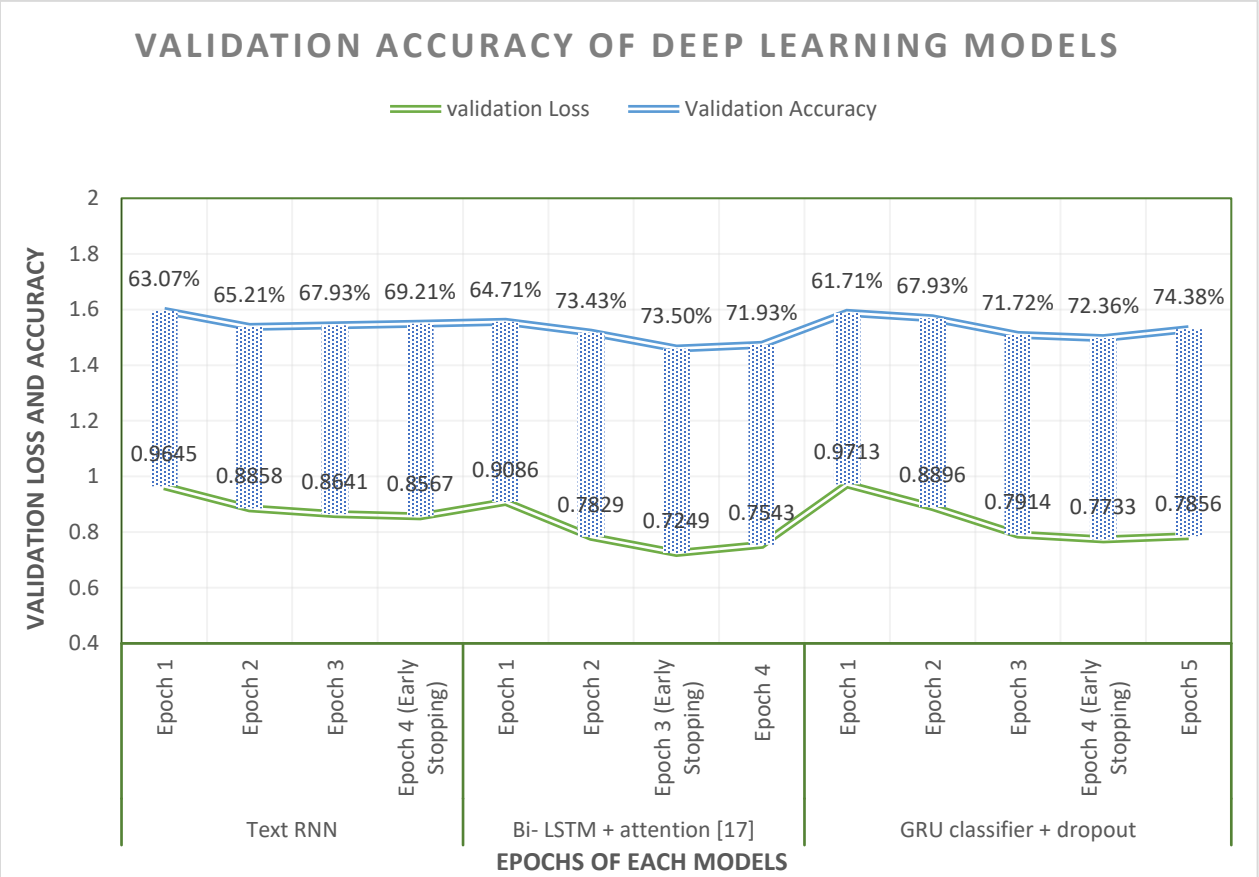

Fig. 9. Deep learning models with validation loss and accuracy. The model was trained with early stopping

Figure 9 shows the gist of the model training process, with the parameters of callback features to early stop the training based on validation loss and categorical cross-entropy, the model was optimized for training and to prevent the overfitting.

6. CONCLUSION AND FUTURE WORK

Text classification is an important problem to be addressed in the real word, as the digitization of the world is ahead, efficient methodologies must be practiced in order to organize the digital documents. This report presents an overview of popular research methodology in text classification which is using deep neural networks for text classification. The experimented methods of RNN and other variants suggested the affluence in handling big amounts of data efficiently. As the data grows bigger there always comes the problem of data sparsity, feature extraction and handling of a sequence of data. Unlike linear classifiers, RNN and its variants can capture the long-term dependencies for language modeling and text classification. Although, linear classification models have the capability to classifying the documents to a maximum extent, their scope is limited, since they fall back in scaling. If the data is huge in number, eventually sparsity problem hinders them to decide on predicting class labels, also other preprocessing steps must be considered before proceeding further.

The methods presented above can handle large amounts of text documents and above results show the overall performance improvements, though it comes at the expense of processing power. Also, there is a scope of improvements in the existing RNN architecture by using more advanced word embeddings like BERT, fasttext to increase accuracy. In addition to this, we will also consider more sophisticated methods for labeling the text corpus for an accurate prediction of class labels as our future works. It is safe to consider that, building a spell checker dictionary around the text corpus before tokenizing the words and proceeding further for embeddings will increase the accuracy. Furthermore, hierarchical deep learning model approach presented in [10] can be explored for this documents, where level1 will be considered as “NATO secret” or “NATO non-secret” and leve2 classes being the current research problem of interest.

7. REFERENCES

[1]. Zhou, Chunting, Chonglin Sun, Zhiyuan Liu, and Francis Lau. "A C-LSTM neural network for text classification." *arXiv preprint arXiv:1511.08630* (2015).

[2]. Zhang, Yin, Rong Jin, and Zhi-Hua Zhou. "Understanding bag-of-words model: a statistical framework." International Journal of Machine Learning and Cybernetics 1, no. 1-4 (2010): 43-52.

[3]. Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In Advances in neural information processing systems, pp. 3111-3119. 2013.

[4]. Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

[5]. Gal, Yarin, and Zoubin Ghahramani. "A theoretically grounded application of dropout in recurrent neural networks." In Advances in neural information processing systems, pp. 1019-1027. 2016.

[6]. Joachims, Thorsten. "Transductive inference for text classification using support vector machines." In Icml, vol. 99, pp. 200-209. 1999.

[7]. Fernández-Delgado, Manuel, Eva Cernadas, Senén Barro, and Dinani Amorim. "Do we need hundreds of classifiers to solve real world classification problems?." The Journal of Machine Learning Research 15, no. 1 (2014): 3133-3181.

[8]. Liu, Pengfei, Xipeng Qiu, and Xuanjing Huang. "Recurrent neural network for text classification with multi-task learning." arXiv preprint arXiv:1605.05101 (2016).

[9]. Lai, Siwei, Liheng Xu, Kang Liu, and Jun Zhao. "Recurrent convolutional neural networks for text classification." In Twenty-ninth AAAI conference on artificial intelligence. 2015.

[10]. Kowsari, Kamran, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. "Hdltex: Hierarchical deep learning for text classification." In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 364-371. IEEE, 2017.

[11]. Tang, Duyu, Bing Qin, and Ting Liu. "Document modeling with gated recurrent neural network for sentiment classification." In Proceedings of the 2015 conference on empirical methods in natural language processing, pp. 1422-1432. 2015.

[12]. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[13]. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780.

[14]. Smith, Ray. "An overview of the Tesseract OCR engine." In Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 2, pp. 629-633. IEEE, 2007.

[15]. Erhan, Dumitru, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. "Why does unsupervised pre-training help deep learning?." Journal of Machine Learning Research 11, no. Feb (2010): 625-660.

[16]. Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532-1543. 2014.

[17]. Zhou, Peng, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. "Attention-based bidirectional long short-term memory networks for relation classification." In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), vol. 2, pp. 207-212. 2016.