ATHABASCA UNIVERSITY

LEVERAGING LOCAL AND GLOBAL WORD CONTEXT FOR MULTI-LABEL DOCUMENT CLASSIFICATION

BY

ROBERT ELLIS

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN INFORMATION SYSTEMS

FACULTY OF SCIENCE AND TECHNOLOGY

SCHOOL OF COMPUTING AND INFORMATION SYSTEMS

ATHABASCA, ALBERTA

OCTOBER, 2020

Approval of Thesis

The undersigned certify that they have read the thesis entitled

LEVERAGING LOCAL AND GLOBAL WORD CONTEXT FOR MULTI-LABEL DOCUMENT CLASSIFICATION

Submitted by

Robert Ellis

In partial fulfillment of the requirements for the degree of

Master of Science in Information Systems

The thesis examination committee certifies that the thesis
and the oral examination is approved

Supervisor:
Dr. Dunwei Wen
Athabasca University

Committee Members:
Dr. Ali Dewan
Athabasca University

External Examiner:
Dr. Ebrahim Bagheri
Ryerson University

November 3, 2020

Dedication

For Peggy, James, Adrian, and Chloe – without your love, support and understanding, this would not have been possible.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Dunwei Wen. His support, advice and guidance throughout the entire thesis undertaking has proven invaluable.

Additionally, I would like to acknowledge and thank the members of my thesis defense committee – Dr. Ali Dewan and Dr. Ebrahim Bagheri – the discussion and feedback regarding my research and thesis was engrossing and insightful, and the final product is better for it.

I would like to thank the faculty of Athabasca University – your feedback regarding my projects and coursework helped refine my research skills.

My thanks to my current and former leadership at London Health Sciences Centre – Cory Gosnell, Jennifer McCallum, Dominic Langley, and Deepak Sharma – your support and understanding through all stages is greatly appreciated.

Abstract

With the increasing volume of text documents, it is crucial to identify the themes and topics contained within. Labelling documents with the identified topics is called multi-label classification. Interdependencies exist between not just words, but sentences and paragraphs. These longer sequences and more complex relationships increase the label identification challenge. Five novel deep neural networks are proposed and evaluated for their performance classifying longer documents. The RCLNN applies the RCL to NLP, combining that model with a CNN which has demonstrated success on short text. The QRCNN similarly extends a CNN in addition to implementing it with a QRNN. The remaining three models build on these base models, integrating them in a novel pseudo-Siamese approach. Experiments find QRCNN highest performing overall, with the PSRCNNA model a close second, indicating that the pseudo-Siamese approach can be performant when married with attention.

Keywords: Recurrent, Convolutional, Neural network, Classification, Attention, Hierarchy, Ensemble, Siamese

Table of Contents

List of Tables

List of Figures and Illustrations

Chapter 1. Introduction

Twitter. News articles. Pathologist reports. Text is pervasive in our lives. As text is transcribed language, it is a foundational method in how we communicate and relay knowledge to each other and is a primary mechanism with which we preserve knowledge for later reference. Text documents are fundamental to our lives.

The volume, velocity and variety of new information is ever increasing (Hilbert & Lopez, 2011; Chen & Zhang, 2014; IBM, n.d.). Given that text is a common format for such information, the number of documents is also increasing similarly. As is our dependence on the knowledge that such documents provide. With increased dependence and usage comes the need to easily identify documents which go together, what topics they include, and other relevant information. In the past, humans have developed systems to support this – a clear example being the Dewey Decimal System (Dewey, 1876) and card catalogues employed at libraries worldwide.

Historically, these documents have been physical in nature – inscribed in some fashion onto paper which is bound together, forming books, magazines, and newspapers amongst other products. With the digital revolution, our documentation has shifted to electronic formats. This shift is what facilitates the ever-increasing onslaught of information, and brings new challenges, but also new opportunities. Where once we needed to be more prescriptive in what is retained, we can now keep versions and variations of documents without number – and retain not just the informative, but the trivial. What was once a manual process to search and identify documents of value can now be automated. The internet has become our library, with the world's knowledge at our fingertips.

Language has nuance and ambiguity; language has topics and subtopics and variability of meaning. As a result, a document is more than the sum of its words, but the interplay between them – which is why context is king. The context of a situation or word or document colours the meaning, interpretation and value of the element in question. It provides additional information helping determine applicable word sense for example or determine the tone of a phrase. In the space of documents, context refers to, and is derived from, the preceding and following words to the component in question.

Traditional natural language processing (NLP) methods are often rule and logic based, with some probability thrown into the mix. Natural language is not necessarily a clean, rules-based construct, but a living entity which has morphed over time and space – clearly seen in the dialects which exist, and how word choice and connotation varies between users across the globe.

Punctuation also aids in understanding. Consider the following two statements:

- "Eat, James!"
- "Eat James!"

The first statement is a directive to James to eat, whereas the second is suggesting an act of cannibalism, with James as the dish – all for want of a comma! Many traditional NLP models, such as the bag of words model, in an effort to simplify the problem disregard punctuation and yet are effective – but would fail to capture this particular nuance without the context the comma brings. To really work with text in all its forms, NLP algorithms need methods of deciphering and encoding context. Clearly, written language is naturally sequential – characters feeding into words, into sentences and longer documents. This structure brings context with it and shapes the context as well. Which

requires considering the sequencing of words and not just simple consideration of word presence alone. Words impact words, and this impact – this context – must be considered appropriately.

As previously mentioned, the need to organize documents in meaningful ways is crucial – as it allows one to narrow down and identify the texts containing the information one is looking for. To achieve this, one must identify themes, topics, and key words. In the paper world, this identification task is done manually. Post-digital revolution, as there was a shift to digital formats from paper, there has been a shift to computational methods over manual for identification of such things. The aforementioned rate of information production has introduced new challenges. Larger data volumes beget the need for processes which can cope with such volume and velocity.

Enter deep neural networks (DNN). DNN have hit a resurgence and present a viable method for addressing the data volumes of today. DNNs at their core learn representations – that is, the DNN identify the key features of their input, and develop encodings which embed these features (Goodfellow, Bengio, & Courville, Introduction, 2016). The DNNs learn to identify these features through training – which requires minimal user intervention. Supervised learning for example does require previously labelled training data – but the specific features do not necessarily need to be called out. Unsupervised learning does not require such labelled data. This automated method for feature identification is part of what provides DNNs with their power. DNNs are also heavily data parallel, allowing them to leverage the processing power of GPUs. Both of these facts alone would position DNNs well for handling the influx of data today. The

performance DNNs have seen, however, makes them the clear choice, as they have been generally out performing traditional methods (Bengio & LeCun, 2007; Najafabadi, et al., 2015; Lai, Xu, Liu, & Zhao, 2015; Allison, Guthrie, & Guthrie, 2006) as well. The learned representations have a variety of utility and facilitate: identifying cats in videos (Le, et al., 2011), self-driving cars (Bojarski, et al., 2016), improved voice recognition (Hinton, et al., 2012) and patient mortality prediction (Grnarova, Schmidt, Hyland, & Eickhoff, 2016).

Text today also has new challenges – as dialects have arisen, so have variations of word usage to suite the tools of the day, many with arbitrary length limits. And with these snippets of text old rules need to be revised, vocabularies expanded. Given the rise of Twitter, messaging applications and phone texting, the volume of this form of text has expanded considerably, which has naturally led to much focus being placed on the application of modern NLP techniques to shorter text (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013; Liu Y. , Liu, Chua, & Sun, 2015; Rush, Chopra, & Weston, 2015; Severyn & Moschitti, 2015). Techniques such as word2vec (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) learn word representations from the word's local context, thus permitting methods to represent new (or custom) "words".

Longer text has its own set of challenges. With more words comes more subtopics, greater complexity of word relationships, and longer sequences to evaluate (Cohen, Ai, & Croft, 2016; Jozefowicz, Wojciech, & Sutskever, 2015). NLP on longer text does benefit from the work on short-text – clearly seen in the adoption of word2vec embeddings (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) to feed networks instead of previously common one-hot vectors. As well, the relationships in short text

exist in longer text as well – just a larger volume and over longer range. Thus, techniques which have shown promise on short text should have some degree of applicability to longer text. Convolutional neural networks (CNN) are one such approach, and while they have been successfully applied to NLP tasks on longer text (Grnarova, Schmidt, Hyland, & Eickhoff, 2016; Liu, Chang, Wu, & Yang, 2017), CNN have been shown to be unable to incorporate longer term dependencies (Cohen, Ai, & Croft, 2016). Recurrent neural networks (RNN) are a separate approach to sequence modelling that better handles longer-term dependencies (Cohen, Ai, & Croft, 2016), and similarly have been successful in NP (Jagannatha & Yu, 2016; Cho, et al., 2014). RNN however have been shown inferior to CNN for fine-grain sequences (Cohen, Ai, & Croft, 2016).

A key NLP challenge is the identification of what a piece of text is "talking about" – that is, what topics and subtopics are present. Longer text has a greater potential to include multiple interwoven subtopics than short text by virtue of the amount of words and sentences present. There is much value in the identification of these subtopics – be it the organization of documents previously mentioned to enabling search engines and grouping documents together. One classic approach to this problem has been topic modelling. Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan, 2003) is a well-established generative approach to this, evaluating documents and constructing probability distributions, allocating words to topics, and topics to documents. There has even been an extension to LDA which incorporates word2vec embeddings – combining the benefits of modern methods with a traditional approach (Das, Zaheer, & Dyer, 2015). The topics found by LDA consist of related words – with the relations drawn from the

documents. These topics are not labelled, and at times require some interpretation to identify what they represent. They may also not align with what one would naturally identify as a topic of the document – e.g. a found topic could consist mostly of numbers.

Subtopics can also be considered features of the document. Which makes them prime candidates for DNNs. One approach to identifying subtopics with DNNs is multi-label classification, where each label can be seen as reflecting a component topic of the document and draws from the entire text. CNNs have been used to approach multi-label classification (Liu, Chang, Wu, & Yang, 2017; Gargiulo, Silvestri, & Ciampi, 2018) but suffer from the previously noted sequence length challenges (Cohen, Ai, & Croft, 2016). Hybrid approaches, integrating RNN and CNN have also been applied to multi-label text classification, attempting to leverage the feature identification strength of CNN with the superior sequencing RNN provides (Chen, Ye, Xing, Chen, & Cambria, 2017), the approach is very much CNN then RNN, not integrating the two different approaches at handling sequences. While the various topics must be "known" in the case of multi-label classification (unlike topic modelling – such as LDA - where the topics themselves are "discovered") – it is not a detriment. Known, interpretable topics – the classes – allows them to be leveraged in an automated fashion – not requiring interpretation of groups of similar words as we have with topic models. Given data volumes – this is crucial to information retrieval. DNNs have also shown to be more proficient than traditional methods for NLP (Bengio & LeCun, 2007; Najafabadi, et al., 2015; Lai, Xu, Liu, & Zhao, 2015; Allison, Guthrie, & Guthrie, 2006).

Research Questions

Given the value of subtopics, the inclusion of subtopic information into learned representations is crucial. The challenges posed by the big data of today make DNN methods more crucial – due to their ability to handle such volumes and their performance. This paper focuses on the problem of incorporating subtopic information into representations learned by DNNs, through the lens of multi-label classification. By improving upon the performance of such classifiers, the underlying embedding has its quality improved. In line with this goal, the particular research questions to be addressed are:

- How can we ensure that learned representations incorporate extended term dependencies as presented by longer text?
- Will the integration of local and global word context, jointly learned using different DNN approaches produce better document classification results than the separate models individually?

Thesis Overview

The remainder of this document is organized as follows. Chapter 2 provides the necessary background, including a review of existing literature. In Chapter 3, the neural network models implemented are detailed, as well as the experimental framework, datasets and methods for evaluation. Chapter 4 details the results of said experiments, and Chapter 5 provides the conclusion and highlights possible future work.

Chapter 2. Literature Review

Classification

We have had the means for computers to interpret text for some time – programming languages. Through the use of context-free grammars, custom syntax, specialized rules and translation into machine code, we can relay instructions to a computer in the form of text. Natural language is not so nicely structured or interpreted.

We tackle the problem of interpreting natural language using methods know as natural language processing (NLP). Using traditional NLP methods, we break text up, identify parts of speech, build probabilistic grammars, and generally find methods of breaking text into logical chunks which can then be consumed, broken down and ultimately interpreted. These methods, while proficient, have some challenge working with the significant volumes of text being generated daily. With the influx of new text in the current time, new methods are needed.

Classification (Russell & Norvig, 2010) is the machine learning task of assigning a label, or class, to the provided input – say labelling books as fiction/nonfiction or identifying pathology reports as related to cancer or not. Classification is a form of supervised learning (Russell & Norvig, 2010), a machine learning technique which utilizes labelled training and test data (unsupervised would identify clusters from the data itself). With supervised learning, the model is evaluated on some training input, the outputs compared to the corresponding expected values, the model is adjusted as appropriate, and training repeats until convergence. Performance of the model is determined by evaluating the model on a test set - which involves examples the model has not been trained on – and reviewing various accuracy measures. One round of training over the entire training dataset is called an epoch.

Multiclass classification is the natural extension of simple classification, where instead of a binary choice, the classifier produces a label from a set of labels which it identifies as best fitting the input. Multi-label classification allows for the provision of multiple labels to the input - e.g. labelling an image as fruit, tree, apple, green. One approach to multi-label classification is one-vs-all (Bishop, 2006), where if there are *n* labels, *n* binary classifiers are trained, and the final output is the union of the *n* classifiers outputs. The output layer of DNN classifier can be considered as one-vs-all, with each output node a separate classifier.

Neural Networks

Artificial neural networks are a machine learning technique that have their roots in early neurological models from 1943 (McCulloch & Pitts, 1943). Conceptually, a neural network is a directed graph where the nodes are a mathematical analogue to the biological neuron which inspired it. These artificial neurons output the value of an activation function whose input is the weighted sum of the neuron's inputs, that is:

$$y = f(Wx + b)$$

where *f* is the activation function, *W* is the weight matrix for the inputs, *x* is the neuron's inputs and *b* is the bias. The activation function *f* is typically non-linear, as this allows the network, to address non-linear problems (Goodfellow, Bengio, & Courville, 2016b) and allows the neural network to be a universal approximator (Hornik, Stinchcombe, & White, 1989; Kurkova, 1992). Typical structure for a neural network is an input layer, one or more hidden layers and an output layer. A DNN is a neural network which has multiple hidden layers. This layering is what gives DNNs their power. With the rise of general-purpose GPU programming DNNs have had a surge of progress, as DNNs are

highly parallelizable and GPUs easily handle large data parallel problems (Najafabadi, et al., 2015; Chen & Lin, 2014). The introduction of cycles into the DNN graph provides us with recurrent neural networks (RNN) (Goodfellow, Bengio, & Courville, 2016g).

Training a DNN is the process through which the parameters of the network are tuned to best model the data. DNNs are typically trained through the use of gradient descent, an optimization algorithm which is guaranteed to find a local minimum of the objective function being optimized (Bengio, Simard, & Frasconi, 1994; Bottou, 2010). For a neural network, gradient descent is calculated through the use of back propagation (Rumelhart, Hinton, & Williams, 1985). A learning rate is applied during training, to control the rate of change to the parameters. Much research has been put into variations of gradient descent (Ruder, 2016), which aim to improve the training of deep neural networks. Adam (Kingma & Ba, 2015), Adagrad (Duchi, Hazan, & Singer, 20111) and RMSProp (Tieleman & Hinton, 2012; Goodfellow, Bengio, & Courville 2016e) are examples of such. Each of these algorithms adapt the learning rate themselves, versus needing to adjust the learning rate outside of the algorithm during training.

The objective function used during training typically determines the cost of errors, and the optimizer (e.g. gradient descent) searches the parameter space to identify the parameter set which minimizes said objective function. One such objective function comes by way of maximum likelihood estimation - the cross-entropy loss (Goodfellow, Bengio, & Courville, 2016d):

$$L = -\mathbf{E}_{x\sim\hat{p}_{\text{data}}}[\log p_{\text{model}}(x)]$$

. Taking the log of likelihood simplifies calculations, and through minimizing the cross entropy we maximize the likelihood.

Another objective function which has relevancy to classification is the Hamming Loss, which is the proportion of false predictions (both positive and negative) to ground truth labels.

$$H = \frac{\sum_{i=1}^{N}(FP_i + FN_i)}{NL}$$

As previously mentioned, the activation function adds a non-linear component to the neurons of a DNN. The activation function also ensures that the output of a neuron is constrained to some range. As DNNs are trained through gradient descent, having activation functions which are differentiable has been viewed as desirable, but in practice there are ways to address - such as returning only one side of the derivative of a non-differentiable point (Goodfellow, Bengio, & Courville, 2016b). The Rectified Linear Unit (ReLU)

$$y = \max(x, 0)$$

is one such discontinuous activation function, as it is not continuous at $x = 0$. It is, however, one of the most common activations functions in use (Ramachandran, Zoph, & Le, 2017), in part as it has facilitated improved training of DNNs and ease of calculation. Another classic activation function is the hyperbolic tan function

$$y = \frac{e^{2x} - 1}{e^{2x} + 1}$$

which, while it constrains values to the range [-1,1], saturates heavily for large values of $|x|$, and has been migrated away from.

A key factor in the training of DNNs is the generalization of the network (Goodfellow, Bengio, & Courville, 2016e), in that the networks parameters are generalized enough to be applicable to a host of examples, and not over fit to the training

data. We refer to the various actions which aim to reduce generalization error as regularization (Goodfellow, Bengio, & Courville, 2016f). One key component of this is the initialization of the DNNs parameters. The traditional method (Bengio, 2009) for initializing DNNs has been found to contribute to the challenges in training deep networks (Glorot & Bengio, 2010), leading to different initialization methods being proposed (Glorot & Bengio, 2010; He, Zhang, Ren, & Jian, 2015). Dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) is another approach to regularization. With dropout, nodes within the DNN are randomly disconnected (i.e. weights set to 0) during training, effectively removing the node from the network during that training step. This has the effect of reducing over fitting. Early stopping (Nowlan & Hinton, 1992; Goodfellow, Bengio, & Courville, 2016f) - that is, halting training before over fitting to the training data occurs, is another commonly used regularization approach.

Deep neural networks

Let us consider flavours of DNNs in common use. A feed-forward neural network is the most basic DNN - which is comprised of one or more dense hidden layers and is acyclic. Feed-forward neural networks have been found to be good generators of word embeddings (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013; Bojanowski, Grave, Joulin, & Mikolov, 2017; Liu Y. , Liu, Chua, & Sun, 2015; Mikolov, Chen, Corrado, & Dean, 2013). Previously, a common method of inputting words into an DNNs through the use of a one-hot vector - a vector where each position represents a word in the vocabulary, with a 1 in the element corresponding to the word being fed, and 0s otherwise. This format demonstrates the curse of dimensionality (Goodfellow,

Bengio, & Courville, 2016d; Bellman, 2003) clearly as vocabulary increases. Word embeddings are a more concise representation and have been found to encode semantic meaning and relationships within (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013; Mikolov, Chen, Corrado, & Dean, 2013). The trailblazer embedding proposed, referred to as word2vec, is actually two variations - skipgram and continuous bag of words (Mikolov, Chen, Corrado, & Dean, 2013). Liu et al sought to improve upon word2vec by incorporating topic data (Liu Y. , Liu, Chua, & Sun, 2015) into the skipgram model, drawn from learned topic embeddings. FastText (Bojanowski, Grave, Joulin, & Mikolov, 2017; Joulin, Grave, Bojanowski, & Mikolov, 2016) is another modification to word2vec which involves the incorporation of subword (n-character-gram) information into the word embedding. Feed forward DNNs have also been investigated for text categorization (Zhang & Zhou, 2006).

Convolutional neural networks

A convolutional neural network (CNN) is another type of DNN, also inspired by biology - this time the visual cortex (LeCun, 1989; Goodfellow, Bengio, & Courville, 2016a). Finding particular success in computer vision, CNNs apply a kernel (or filter) to the input, processing the input in patches. Convolution can be conceived as sliding the kernel over the input, taking in a subset at each step, determined by the width of the kernel. In the NLP domain, if the input is words, a one-dimensional kernel would be operating on successive n-grams, convolving words together. Through convolving consecutive *n*-grams, CNNS integrate local-context features into the learned representations. A given convolutional layer will typically be comprised of multiple kernels. CNNs are proficient at dimensionality reduction and teasing out key features

regardless of location (Goodfellow, Bengio, & Courville, 2016a; Yin, Kann, Yu, & Schutze, 2017).

CNNs have demonstrated viability for a number of NLP tasks. CNNs have shown utility when working with shorter text (Kim, 2014; Zhang & Wallace, 2015; Severyn & Moschitti, 2015; Zhao & Wu, 2016; Hughes, Li, Kotoulas, & Suzumura, 2017; Zhang, Henao, Gan, Li, & Carin, 2018). In (Kim, 2014) a single layer CNN with max pooling was applied to the problem of sentence classification. This particular model provides the basis for much subsequent research, including a variation which utilized multiple filter sizes (Zhang & Wallace, 2015). This variation of filter sizes provides a means of varying context windows around words and has potential to strengthen the local context of words when building the representation for the entire document, which will be explored in this paper. A similar model was applied to Twitter sentiment analysis, with a novel approach to weight initialization (Severyn & Moschitti, 2015). Yet another flavour of this model was applied to medical sentence classification – this time involving multiple banks of filters and max pooling (Hughes, Li, Kotoulas, & Suzumura, 2017). A different approach, which first uses an attention layer to first identify context vectors for words, which are then fed to the convolutional layer achieved performance comparative to recurrent neural networks (Zhao & Wu, 2016). In all cases, word embeddings were leveraged, and max-pooling was applied to form sentence representations, which were then fed to a final classifier layer using softmax.

Clearly, CNNs have some capacity to derive meaningful sentence embeddings. This fact led to the integration of a CNN with residual connections - which learns sentence embeddings - and a restricted Boltzman machine classifier, as proposed in

(Zhang, Henao, Gan, Li, & Carin, 2018). The CNN used in the model follows from (Kim, 2014). The resulting model was used to classify medical notes, with reasonable success. Liu et al. propose three CNN models for the classification of longer text. The primary idea underlying each of these models is the subsampling of text from the document to reduce number of words, and convolution with max-pooling to build representations for the sampled chunks of text (Liu L. , et al., 2018). The authors found that good results could be achieved with 10% of the document. Grnarova, et al train a two-layer CNN on healthcare provider notes with an aim to predict patient mortality (Grnarova, Schmidt, Hyland, & Eickhoff, 2016). Their model incorporates external information (provider category) alongside the learned sentence embedding to provide further guidance to the CNN. Likewise, hierarchical CNN, with the addition of attention have been applied to question/answering on longer text (Yin, Ebert, & Schutze, 2016). As well, the authors introduce the idea of "attention pooling" (Yin, Ebert, & Schutze, 2016) - that is, using attention to identify salient sentences, and then pooling the top *n* sentences with highest attention.

CNNs have been explored for the purposes of extreme multi-label text classification (Liu, Chang, Wu, & Yang, 2017; Gargiulo, Silvestri, & Ciampi, 2018) with some success. Both CNN models follow a similar structure - a convolutional layer followed by max-pooling and a dense layer prior to the output. While both networks leverage a different number of filters and general embedding dimension, (Liu, Chang, Wu, & Yang, 2017) leverages dimension reduction whereas the internal dimension is constant in (Gargiulo, Silvestri, & Ciampi, 2018). As well, character level CNNs have been evaluated for classification (Koomsubha & Vateekul, 2017; Zhang, Zhao, & LeCun,

2015) of longer documents. The input for these models is one-hot encodings for the letters – in comparison to the word embeddings leveraged by previous models.

Each of these models showcase the flexibility of CNNs in the NLP domain, and how they can be applied both to shorter- and longer text, to extract the key contextual features of the text and generate powerful representations. A key facet of this capability of CNNs which bears some discussion is the concept of pooling. Pooling (Goodfellow, Bengio, & Courville, 2016a) layers consider a region of their input, and replace it with a new representative value. For example, max-pooling (Zhou & Chellappa, 1998) replaces the values being pooled together with the largest value provided. Other options exist, including mean pooling (where the values are replaced by their mean), and the aforementioned attention pooling, amongst others. Pooling provides a degree of translational invariance (Goodfellow, Bengio, & Courville, 2016a) (useful for flagging that a feature exists, regardless of location), and provides a mechanism for the network to consume input of variable size.

Recurrent neural networks

The need to handle time series and sequences led to the creation of RNNs (Goodfellow, Bengio, & Courville, 2016g). In an RNN, the output of a given layer is fed back into the layer, alongside the input. RNN maintain an internal state, which provides them with a form of memory (Goodfellow, Bengio, & Courville, 2016g), facilitating the handling of longer sequences. Research has shown RNNs to be Turing complete (Siegelmann & Sontag, 1995). It is common to "unfold" the RNN for the length of the sequence during training (Goodfellow, Bengio, & Courville, 2016g), to facilitate the back propagation of gradients. RNNs can be challenging to train. The parameter sharing,

and depth induced by the recurrent connects leads to RNNs experiencing vanishing and/or exploding gradients during training (Bengio, Simard, & Frasconi, 1994; Pascanu, Mikolov, & Bengio, 2013; Bengio, Frasconi, & Simard, 1993), where the values for the activation functions saturate, resulting in gradients which become so small they disappear. Variations of RNNs have been derived to address this – such as the long short-term memory (LSTM-RNN) (Hochreiter & Schmidhuber, 1997) and gated recurrent unit (GRU-RNN) (Cho, van Merrienboer, Bahdanau, & Bengio, 2014; Cho, et al., 2014), which utilize gates on the input, output and internal state to address issues in training, particularly vanishing gradients (Pascanu, Mikolov, & Bengio, 2013; Bengio, Simard, & Frasconi, 1994). The gates control the integration of internal state and input, in essence controlling how long something is remembered by the DNN. GRU- and LSTM-RNN have become leading flavours of RNN.

A common approach to leveraging RNN with text is through the use of a bidirectional RNN (Schuster & Paliwal, 1997) – that is, an RNN which processes the sequence both forwards and backwards (often through the use of two related RNN layers). Such an RNN was proposed for the purposes of classifying text documents (Du & Huang, 2018), and integrates attention into the bidirectional LSTM-RNN to highlight salient words. The hierarchical attention network (Yang, et al., 2016) leverages a similar structure – bidirectional GRU-RNN with attention, however their approach applies attention to the individual hidden states, compared to (Du & Huang, 2018) where attention considers all intermediate representations at one time. The hierarchical attention network also leverages two bidirectional layers building word and sentence representations before classification – whereas (Du & Huang, 2018) is one layer.

Another approach is to use an ensemble of bidirectional LSTM-RNN with attention (Zhou, Zhang, & Wu, 2018), where each DNN involved has a vote. In each case, the consideration of text both forwards and backwards builds a better representation as it considers the entire sequence (Goodfellow, Bengio, & Courville, 2016g). Hierarchical structures provide an interesting mechanism for integrating word context – mirroring the structure of text, building sentence representations from word representations, and document representations from sentence embeddings. A variation of this approach is explored in this paper.

While a bidirectional LSTM-RNN is one way for multiple LSTM-RNN layers to work in concert, another approach is the multi-task learning approach in (Liu, Qiu, & Huang, 2016) used for text classification. In two of the models defined, each task has an assigned LSTM-RNN layer, with linkages between the two layers – feeding outputs from each layer into the other to facilitate the transfer of features learned from each layer's respective tasks. One of the models simply links the two layers together, whereas the third places a bidirectional LSTM-RNN to facilitate the layer linking. A third model was also put forth, which leveraged a single LSTM-RNN for both tasks being learned.

Text is inherently sequential – sequences of letters form words, which form sentences and ultimately documents. Thus, RNNs are a natural fit for working with text, and have seen much success. As previously discussed, CNNs handle sequences as well, though on a generally smaller scale. One problem which can be seen as a sequence modelling problem is named entity recognition (NER). Lyu et al apply a LSTM-RNN to the problem of NER for on sentences from medical text (Lyu, Chen, Ren, & Ji, 2017). Their proposed model leverages both word and character embeddings, which are fed to a

bidirectional LSTM-RNN. The DNN uses conditional random fields instead of the typical softmax for the final classification layer. Jagatha and Yu similarly evaluate LSTM-RNN and GRU-RNN models on longer text (clinician notes), to classify the documents based on the medical events contained within (Jagannatha & Yu, 2016). In both cases, the models exceeded performance of classical methods for these tasks.

The GRU-RNN was proposed as a modification of the LSTM-RNN, and initially used for machine translation (Cho, et al., 2014). The authors constructed an encoder-decoder structure from the GRU-RNN, to translate English text to French. Encoder-decoders perform much like the name suggests - encoding the input into some representation, and then decoding that representation into some output (Goodfellow, Bengio, & Courville, 2016c). A related model to encoder-decoder approaches is the sequence to sequence model (Sutskever, Vinyals, & Le, 2014). Chopra et al propose an encoder-decoder, throwing in attention on the encoder, for sentence summarization (Chopra, Auli, & Rush, 2016).

Recurrent convolutional neural networks

Context is king. Both CNN and RNN address the problem of context in different ways. CNNs evaluate context in the small - over a small window, identifying salient features in that limited range, and are a key approach to feature selection. RNNs conversely consider longer sequences, and thus are empowered to identify and retain longer term dependencies due to their nature. The integration of both approaches is therefore appealing, bringing the combination of small- and larger-scale context considerations together to build a better model - feature identification and localized context of CNN type sequences, alongside the longer sequences, and broader context

which RNNs bring. It is therefore not unsurprising that recurrent convolutional neural networks (RCNN) have been considered and applied to a variety of domains, particularly as the integration of recurrent connections alongside convolutional filters makes the RCNN "more neurobiologically realistic" (Spoerer, McClure, & Kriegeskorte, 2017).

There have been a few different strategies to what has collectively been called RCNN. One approach is to stack convolutional and recurrent layers feeding one into the other (Ushio, Shi, Endo, Yamagami, & Horii, 2016; Wang, Jiang, & Luo, 2016; Donahue, et al., 2015; Kalchbrenner & Blunsom, 2013; Vu, Adel, Gupta, & Schutze, 2016; Wen, Zhang, Luo, & Wang, 2016; Liu L. , et al., 2018; Chen, Ye, Xing, Chen, & Cambria, 2017). This method harnesses the feature extraction capabilities of CNN and sequences via the RNN. The utility of such an approach has been demonstrated in both the computer vision and NLP domains. Donahue et al. apply such a model to the CV tasks of activity recognition and image/video description in videos (Donahue, et al., 2015). The input (a video in this case) is fed into a convolutional layer so as to extract features from the input scenes which are then fed to a LSTM-RNN layer to accommodate sequence learning. Similarly, Wang et al apply a similarly structured model for sentiment analysis of short text in (Wang, Jiang, & Luo, 2016). Specifically, the authors leverage convolution and pooling to extract features from the input sentence, which is then (like (Donahue, et al., 2015)) fed to an RNN layer to address the sequencing of words in the sentence. The model variants put forth by the authors outperformed the reference models on the Movie Reviews and Stanford Sentiment Treebank. In (Ushio, Shi, Endo, Yamagami, & Horii, 2016), the authors propose a recurrent convolutional model for speech which leverages two parallel LSTM-RNN layers (one to capture attributes of

speech, one for speech acts) to model the sequence of utterances, which then feed a convolutional layer to construct sentence embeddings. Kalchbrener and Blunsom use a RCNN to create a discourse model (Kalchbrenner & Blunsom, 2013). The DNN uses a hierarchical CNN component to construct sentence representations over words, and then feeds these to an RNN segment to address discourse. Chen et al use a CNN to extract features from text, which are then fed to an RNN for label prediction (Chen, Ye, Xing, Chen, & Cambria, 2017). Wen et al add highway networks (Srivastava, Greff, & Schmidhuber, 2015) into the RCNN (Wen, Zhang, Luo, & Wang, 2016) and apply the network to sentiment analysis on IMDB reviews. A slightly different approach is taken in (Vu, Adel, Gupta, & Schutze, 2016), where the CNN and RNN components are trained separately, and work in sequence - instead of being trained as a single network. Again, the model was evaluated on sentence text, for the purposes of relation classification.

Another approach is the concept of the recurrent convolutional layer (RCL) put forth by Liang and Hu (Liang & Hu, 2015), which merges recurrent connections, feed forward connections and convolution into a single layer. As with other RNN, the RCL can be unfolded for a predetermined number of time steps, and the recurrent convolutional neural network (RCNN) proposed stacks multiple RCLs together. In their paper, the authors apply the RCNN object recognition, with good results. Liang et al. also applied RCNN to the problem of scene labelling (at multi-scale) in (Liang, Hu, & Zhang, 2015). Spoerer et al. further consider the model from (Liang & Hu, 2015) for occluded object recognition. Wang and Hu extend the model proposed in (Liang & Hu, 2015) by adding gates, operating on the recurrent and feed forward information, similar to LSTM-

RNN and GRU-RNN (Wang & Hu, 2017). Applying their model to scene text recognition, the authors find good results. A different take inspired by the RCL is proposed in (Shin, Kim, Yoon, & Jung, 2018), where the recurrent connections are placed between the convolutional components. Liang and Hu's RCL has shown viability in computer vision approaches but has yet to be applied to NLP. The approach provides an intriguing approach to identifying context, which will be explored in this paper.

Lai et al. propose a different take on the idea of the RCNN (Lai, Xu, Liu, & Zhao, 2015). What the authors refer to as the "recurrent structure (convolutional layer)" (Lai, Xu, Liu, & Zhao, 2015) is really a bidirectional recurrent neural network, with no convolution operation. This RNN layer identifies the left and right context for a given word – thus the portion of the document which precedes the word, as well as the portion which follows, are summarized. These context vectors are then concatenated to the word itself and fed to a dense feed-forward layer. The portion taken from CNN is the max-pooling layer, which pools these context-filled word representations, to construct a final representation for the document. The authors applied their RCNN to text classification – with good results. The approach taken is interesting, as it identifies the context of a word in relation to where it is in the text, thus providing a comprehensive global context for the word. Integrating this approach with more traditional CNN-type components could provide a mechanism to strengthen the local word context and is explored later in this paper.

Yet another approach is that taken by Yang, who proposes replacing the typical convolutional filter with one constructed with a RNN, creating what is being referred to as a recurrent neural filter (RNF) (Yang, 2018), which can be used as a drop-in

replacement for traditional CNN filters. The RNF considers a sequence of length *w* (the CNN filter window length) and returns the last state of the RNN as the filter value (Abadi, et al., 2016). The author indicates that the motivation for the RNF is to address a failing of traditional linear CNN filters - the inability to "account for language compositionality" (Yang, 2018).

Where (Yang, 2018) implements a convolutional filter using an RNN, the quasi-recurrent neural network (QRNN) (Bradbury, Merity, Xiong, & Socher, 2017) conversely implements an RNN with convolutional filters. Additionally, convolutional filters are used for gate implementation, which has parallels to LSTM-RNN gates. The gate outputs are integrated through variations of dynamic average pooling (Balduzzi & Ghifary, 2016), depending on which gates are necessary. For example, what the author's name *fo*-pooling, combines the values of the output and forget gates. The primary motivation behind the QRNN is performance - both in run time as well as the standard accuracy improvement goals. As the significant usage of convolutional filters increases the capability for parallelism, the performance goal is achieved. Additionally, on the NLP tasks which the authors test the QRNN, the QRNN outperformed equivalent LSTM-RNN. The QRNN is designed as a drop-in replacement for RNNs, similar to RNF for convolutional filters. The integration of CNN and RNN of QRNN presents an interesting approach, integrating the local context identified by the CNN filters, with the more global sequencing context of an RNN. QRNN provides a good foundation on which the proposed models later can build.

Clearly, the integration of CNN and RNN have proven viable in a number of applications, including NLP. The stacking method adopted by (Ushio, Shi, Endo,

Yamagami, & Horii, 2016; Wang, Jiang, & Luo, 2016; Donahue, et al., 2015), while simple has been effective. As each component evaluates sequences differently, the stacked approach can be seen as integrating local and global context, to build feature representations which encompass more than the element itself. The RCL of (Liang & Hu, 2015; Liang, Hu, & Zhang, 2015; Spoerer, McClure, & Kriegeskorte, 2017; Wang & Hu, 2017) takes the integration a step further, intertwining feature extraction within the sequence evaluation, in a way building local context within the broader context of the sequence, instead of layering the context approach. Both (Yang, 2018) and (Bradbury, Merity, Xiong, & Socher, 2017) take a very different approach, instead re-implementing key components using the alternate network type. And (Lai, Xu, Liu, & Zhao, 2015) uses bidirectional RNNs to build anterior and posterior context, simulating CNN behaviour with an RNN, and leveraging pooling. Each are very different but build something more performant than the parts individually.

Additional DNN concepts

Another approach to addressing exploding/vanishing gradients is the inclusion of skip connections (also known as residual connections). These connections bypass a collection of non-linear layers, carrying their source values forward (He, Zhang, Ren, & Sun, 2016). Additionally, skip connections have been found to speed up learning.

Attention is another aspect of the mind which has been emulated after a fashion in DNN. In the living brain, attention is the ability to focus on some sensory input. In a DNN, attention mechanisms provide a means of weighting input - emphasizing what is determined to be important, and deemphasizing the unimportant, often through the use of softmax. Attention has been found to be particularly effective in translation (Bahdanau,

Cho, & Bengio, 2016; Luong, Pham, & Manning, 2015; Vaswani, et al., 2017), sentence summarization (Chopra, Auli, & Rush, 2016; Rush, Chopra, & Weston, 2015), document classification (Yang, et al., 2016) and natural language inference (Parikh, Tackstrom, Das, & Uszkoreit, 2016). In the case of translation, the attention mechanism helps in aligning words between the input and output (Bahdanau, Cho, & Bengio, 2016; Luong, Pham, & Manning, 2015; Vaswani, et al., 2017). In particular, Vaswani et al propose a new model for translation, the transformer (Vaswani, et al., 2017), which unlike typical DNN approaches to this problem, does not leverage recurrent or convolution components. In the realm of document classification, a hierarchical DNN was proposed, leveraging attention between the word and sentence RNNs, and between the sentence RNN and output layer (Yang, et al., 2016). Self-attention (Parikh, Tackstrom, Das, & Uszkoreit, 2016; Zhang, Goodfellow, Metaxas, & Odena, 2018; Cheng, Dong, & Lapata, 2016) allows the network to emphasize salient information within a sequence. This ability to emphasize / de-emphasize may provide for improved identification of local context within documents and is explored in the later research of this paper.

Ensemble methods reliably improve generalization by leveraging a strategy of model averaging (Goodfellow, Bengio, & Courville, 2016f). The motivation being the idea that different models have different approaches, and will produce dissimilar errors for a given input (Perrone & Cooper, 1992; Krogh & Vedelsby, 1995; Dietterich, 2000), thus through aggregating the responses a more general response is determined.

Siamese networks (Bromley, et al., 1993) are DNNs, consist of two identical component networks, joined at the outputs. The use of Siamese networks has been found to be quite effective of for similarity determination - be it signature comparison

(Bromley, et al., 1993), text similarity (Neculoiu, Maarten, & Rotaru, 2016), enhancing word embeddings (Kenter, Borisov, & De Rijke, 2016) or identifying similar questions (Das, Yenala, Chinnakotla, & Shrivastava, 2016). Siamese networks accomplish this through computing representations of the inputs and evaluating a similarity metric (e.g. distance between them). Varying the approach of Siamese network somewhat, by leveraging two *different* DNN component models instead of *identical* components (as is typical) would provide a means to merge these different model's learned representations. Merging such representations for NLP could facilitate integrating local and global context of the words and bring better representations.

As has been identified, multi-label classification is a crucial problem today, only growing in importance with the volume of text being generated. DNN provide a viable, perhaps necessary, means of classifying data at such volumes. CNN and RNN provide different approaches to sequence modelling, and thus of teasing out the context and subtext of documents. Given their individual benefits, there has been research into integrating these approaches, constructing so-called recurrent-convolutional neural networks. The approaches to this integration have been varied, but shown viable. Other methods of integrating DNN content come in the form of Siamese and hierarchical approaches. To more completely represent extended term dependencies there is a need to consider both the local and global context of words in the document. CNN and RNN both provide viable approaches that naturally look at different scales of sequences, suggesting that the integration of these approaches will better integrate the aforementioned context types. As well, different models provide different perspectives (i.e. different representations) and potentially identify different features. Integrating

these perspectives, similar to integrating RNN and CNN, through a Siamese-like approach may provide for better representations.

Chapter 3. Method

Model Overview

As previously discussed, multi-label classification of longer text is an important problem, given the proliferation of such text and the need to organize and search these documents. To address this problem, five new DNN models are proposed, implemented, and evaluated. Specific details pertaining to the models is below.

Given that the base models are integrated together, in either Siamese-like or hierarchical fashion, to build further models, some modification is needed to facilitate the integration – that is, when models are combined into a larger model, the input layer and final stages are adjusted accordingly so as to only have one of each. These are clearly identified in the diagrams as the dashed line components.

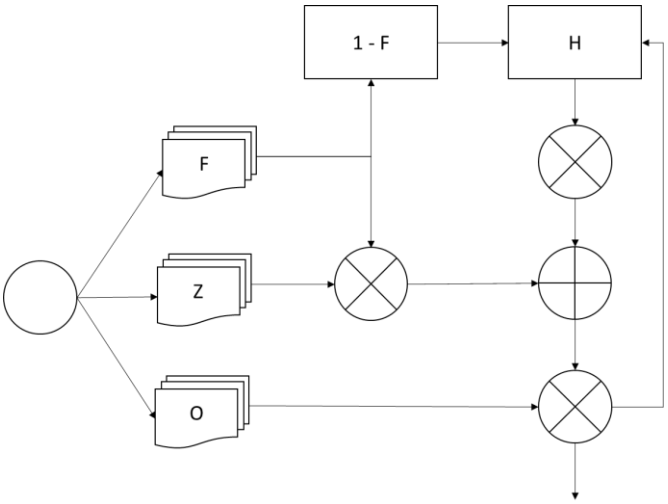
QRCNN

The Quasi-Recurrent Convolutional Neural Network (QRCNN) model is a modification on the RCNN proposed in (Lai, Xu, Liu, & Zhao, 2015), which has been shown performant at multiclass classification of text. A key feature of this RCNN is the context vectors which are constructed for each word, encompassing the entire left and right context respectively for the word. These context vectors then aid in generating a word representation which entails the words context as it lies within the entire document. While the authors in (Lai, Xu, Liu, & Zhao, 2015) evaluated the RCNN against multiclass classification only, the model has potential for multi-label classification as it integrates comprehensive global context into each intermediary embedding. The primary shift required to support multi-label classification is the use of the logistic function in the output layer, versus the softmax leveraged by the authors.

QRCNN implements the bidirectional RNN component of RCNN using QRNNs (Bradbury, Merity, Xiong, & Socher, 2017) variant instead of traditional RNN or LSTM-RNN. The structure of QRNN is shown in *Figure 1*. QRNN is considered to be a drop-in replacement for RNN (Bradbury, Merity, Xiong, & Socher, 2017). As the QRNN is implemented in Python and Tensorflow, the speed improvement available if implemented natively does not manifest, but operational performance is retained. Z, F, O and H are the candidate embedding, forget gate, output gate and hidden state respectively. *Figure 2* details the structure of QRCNN. Other changes QRCNN brings to RCNN is the addition of two dense layers (dashed boxes labelled "Dense").

Figure 1

QRNN



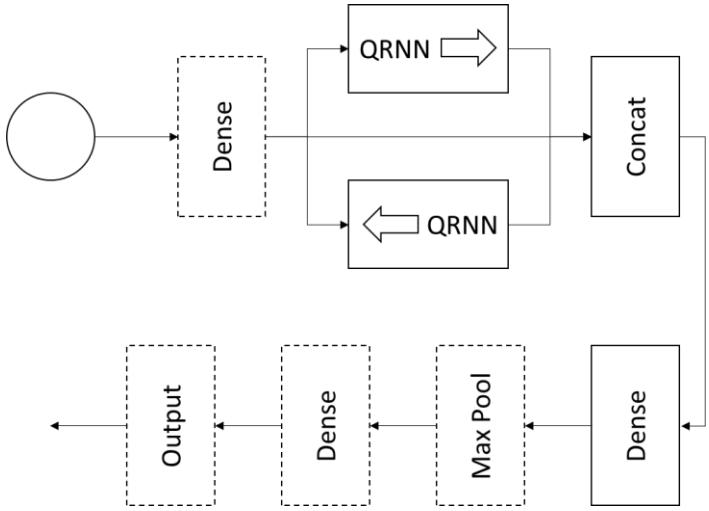
Note: The ⊗ denotes multiplication, ⊕ denotes addition. F, Z & O are 300 width 2 1D convolutions.

The primary activation function in this implementation of QRNN has been changed to RELU instead of the sigmoid function used in the original implementation (and is not applied element-wise for F, O). The use of a QRNN and RELU in this instance was

primarily motivated to manage runtime errors arising within Tensorflow when RNN and LSTM-RNN were utilized. Having said that, LSTM-RNN itself arose as a means of addressing the vanishing and exploding gradients which arise with traditional RNN (Hochreiter & Schmidhuber, 1997).

Figure 2

QRCNN



Note: Dashed outline identifies components excluded when part of a larger model. QRNN arrows denote the bidirectional components.

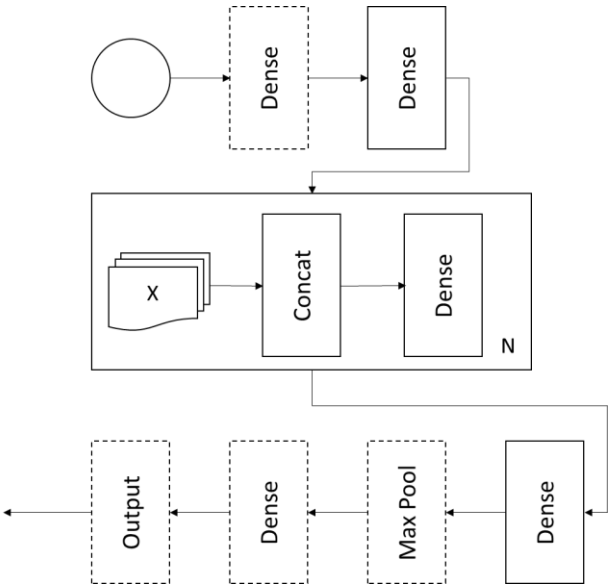
RCLNN

Recurrent Convolutional Layer Neural Network (RCLNN) is a RCNN with one RCL (Liang & Hu, 2015), leveraging an internal convolutional structure derived from the CNN in (Zhang & Wallace, 2015). The RCL has proven viable in computer vision. The intent with choosing this model is to evaluate its utility in NLP. The RCL structure facilitates looking for features at different levels of aggregation – this can be visualized as a step-pyramid structure, with the greatest level of detail (the raw input) as the base, and each successive layer repeating the same evaluations on the previous layers results,

in essence amplifying features found. In the context of NLP, this would be looking for context features at different scales of context window.

Figure 3

RCLNN



Note: Plate notation used for the RCL, which is unrolled n time. Dashed outline identifies components excluded when part of a larger model. *N* denotes the recurrent window size, and *X* is the set of convolutional filters.

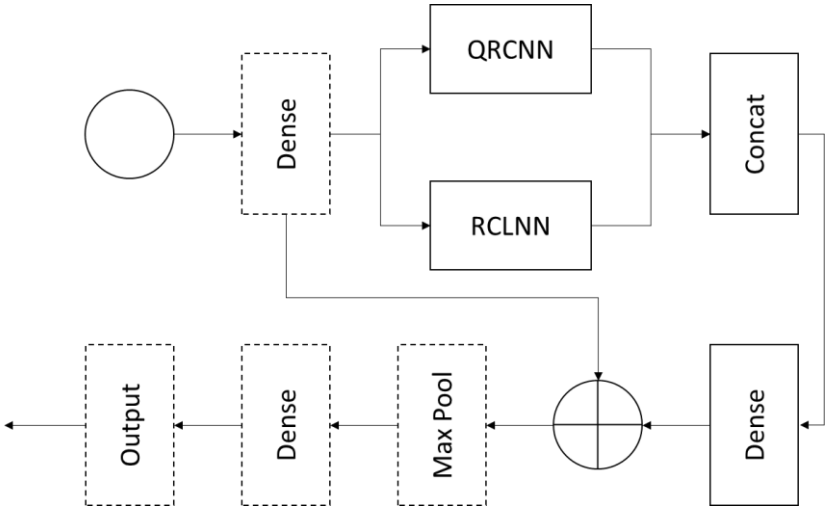
The inner convolutional structure follows from (Zhang & Wallace, 2015), a known model which has been applied for sentence classification. In my implementation, the final dense layer's softmax has been replaced instead with RELU (as the model is not performing classification directly). As well, the convolution sizes vary. *Figure 3* showcases the structure of RCLNN, using plate notation for the RCL portion. *N* is the depth of the recurrent window, and *X* is the set of convolutional filters applied to the input.

PSRCNN

Pseudo-Siamese Recurrent Convolutional Neural Network (PSRCNN) is a new model, forming a Siamese-like ensemble of QRCNN and RCLNN, with a common input layer, and where the outputs of QRCNN and RCLNN are concatenated, and then fed to the output layers. The structure of PSRCNN is detailed in *Figure 4*.

Figure 4

PSRCNN



Note: ⊕ denotes addition; QRCNN is described above in Figure 2 (solid line), and RCNN in Figure 3 (solid line)

In a traditional ensemble setup, multiple DNNs provide predictions with the final output of the ensemble being a determined from these – similar to casting votes and majority rules (Perrone & Cooper, 1992; Krogh & Vedelsby, 1995; Dietterich, 2000). The power in ensemble models is the ability to leverage different approaches – different perspectives if you will – in determination of the final result. In the instance of PSRCNN, instead of simply averaging the results, the results are combined through

concatenation into a new aggregate representation, which is then fed to a fully connected layer for further integration.

Siamese models are an approach typically used for similarity evaluation. Two DNNs are configured and share a common output layer which evaluates a similarity metric. In PSRCNN, the two component models are different from each other, and joined at both the input and output layers (hence pseudo-Siamese). Instead of evaluating the similarity between the sub-representations, the representations are integrated rather.

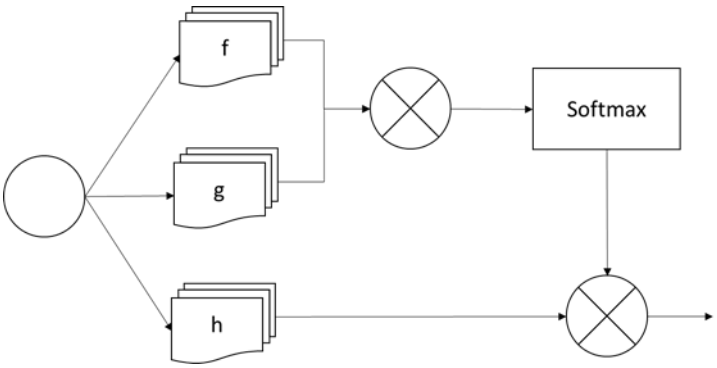
The intent with the PSRCNN model is to leverage the power that an ensemble or Siamese approach brings – integrating two models into one larger model. The representations learned by PSRCNN thus incorporate the global context of QRCNN with the long-term local context of RCNN. A skip connection (He, Zhang, Ren, & Sun, 2016), which integrates the value of the initial dense layer with the learned integrated representation is included to speed up learning and address gradient issues.

PSRCNNA

The Pseudo-Siamese Recurrent Convolutional Neural Network with Attention (PSRCNNA) incorporates a self-attention layer between the dense layer and output layers of PSRCNN. The self-attention implemented here (outlined in *Figure 5*), follows from (Zhang, Goodfellow, Metaxas, & Odena, 2018). Attention has been found to aid in identify salient features – and should improve the quality of learned representations, by highlighting the crucial features. Inclusion of both PSRCNN and PSRCNNA permit the evaluation of the impact of self-attention on the learned representations. The structure of PSCRNNA is detailed in *Figure 6*.

Figure 5

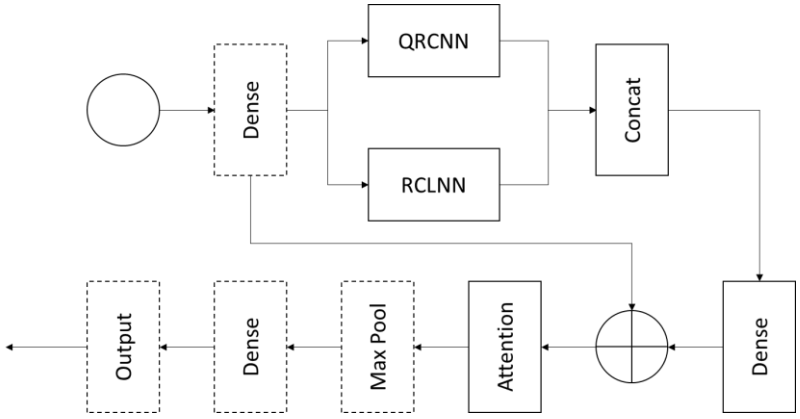
Self-Attention



Note: The ⊗ denotes multiplication; *f* and *g* are 5 1x1 convolutions used to calculate attention; *h* is 300 1D convolutions bringing the input into the feature space.

Figure 6

PSRCNNA



Note: Attention is detailed in Figure 5. ⊕ denotes addition.

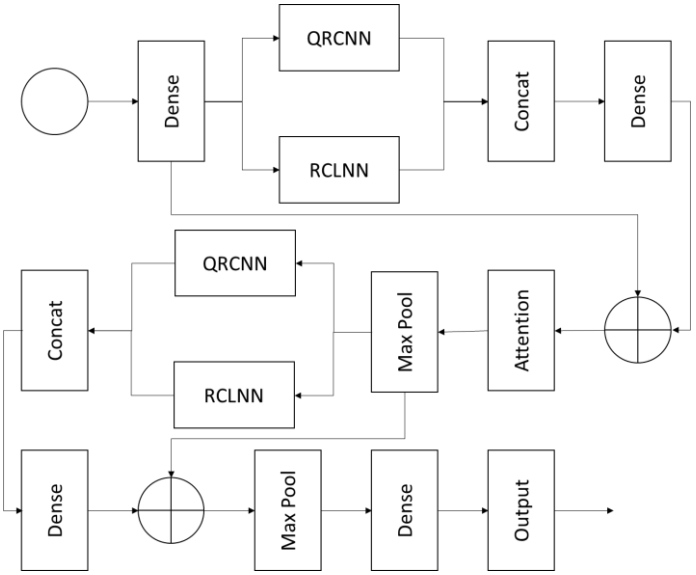
HPSRCNN

The Hierarchical Pseudo-Siamese Recurrent Convolutional Neural Network (HPSRCNN) model introduces hierarchy into the mix, in line with the hierarchical attention model (Yang, et al., 2016) and the structure of language. In particular, as a document is formed of sentences, which themselves are formed of words, the HPSRCNN

model takes the word representations generated by a PSRCNNA component, pools these embeddings to form sentence representations, which are then fed to a PSRCNN structure for determination of the final document embedding. The entire structure forms one complete, hierarchical DNN as seen in *Figure 7*. HPSRCNN provides a mechanism to explore if such hierarchical integration provides improvement when used in conjunction with the proposed PSRCNN and PSRCNNA models.

Figure 7

HSPRCNN



Note: QRCNN, RCLNN denoted in Figures 2 and 3 respectively (solid line); Attention detailed in Figure 5; ⊕ denotes addition

Implementation Details

More concrete implementation details for the models are as follows. In general, layers are initialized using Glorot uniform initialization (Glorot & Bengio, 2010) and use RELU (Ramachandran, Zoph, & Le, 2017) as the activation function. DNN input is

word embeddings of dimension 300. Likewise, the dimension of the outputs for each layer is 300. Max pooling is used to build document (and sentence in HPSRCNN) representations. Dense layers (other than the input and output layers) apply dropout with a factor of 0.5. For the final classification output layer, the logistic function is used, and has N outputs – where N is the number of classes.

As previously stated, QRCNN is an implementation of the recurrent convolutional neural network defined in (Lai, Xu, Liu, & Zhao, 2015), using a QRNN (Bradbury, Merity, Xiong, & Socher, 2017) instead of a traditional RNN. The QRNN uses fo-pooling, and a 1D convolution window size of 2 (and so mimics traditional LSTM behaviour (Bradbury, Merity, Xiong, & Socher, 2017)). RELU is used for the convolution activation function. The left and right context vectors are initialized from a uniform distribution. The final dense layer of QRCNN however uses the TANH activation function.

RCLNN performs dimension reduction using an initial dense layer (reducing dimension from 300 to 90). The recurrent convolutional layer is manually unrolled (4 steps when working with word embeddings, and 2 steps when working with sentence embeddings) and contains 90 1D convolutional filters (30 each of sizes 1/3/5 for word embeddings, and sizes 2/3/5 for sentence embeddings). The filters apply dropout with a factor of 0.2, and max pooling. Finally, a dense layer to return the output of RCLNN to dimension 300.

For the PSRCNNA and HPSRCNN DNN models, the format of the self-attention layer implemented is taken from (Zhang, Goodfellow, Metaxas, & Odena, 2018), with dropout added. f and g comprise of 5 convolutional filters each, are 2D, with a window

size of (1,1) and apply dropout at 0.2 and max pooling. *h* consists of 300 1D convolutional filters with window size of 1 and applies dropout at 0.2. Gamma is initialized to 0 and is learned during training.

Learning uses the Adam (Kingma & Ba, 2015) optimizer, with epsilon set to 0.001, using log loss. The learning rate is 0.0001. Document sizes are set to a maximum of 500 words, and sentence length is considered to be 10 words. Meta-parameters were adjusted through a manual grid search. Mini-batches of size 64 documents (32 per graphics card) were used. Primary considerations for the meta-parameters were memory management, gradient management and training time. RMSProp (Tieleman & Hinton, 2012) was considered alongside Adam, but as both performed similarly during evaluation, Adam was adopted.

The DNNs were implemented using Tensorflow 1.10 (Abadi, et al., 2016), and experiments were performed on a Linux PC with 48 GB RAM, Quad-core 3.4 GHz i7 processor, and dual nVidia GeForce 1080 graphics cards.

Datasets

Each of the base datasets chosen for the experiments are established in the literature. The details for the datasets used are below, with some highlights in Table 1. Cardinality is the average number of labels per elements of the dataset, and density is the cardinality divided by the number of labels (Tsoumakas, Katakis, & Vlahavas, 2009), with research suggesting that low label density can impact the F_1 score and other measures (Bernardini, da Sliva, Rodovalho, & Meza, 2014). Each dataset was chosen for its availability with multi-label classification, to represent different types of text, and to

vary label density. As we see in Table 1, the label mixture between the test and training sets is relatively well matched.

Table 1

Dataset Features

Dataset	Train Size	Test Size	Labels	Vocab	Train Density	Train Card.	Test Density	Test Card.
Amazon	8155	292	8	34724	0.196	1.564	0.200	1.599
Enron	1686	128	10	37582	0.371	3.712	0.402	4.016
Reuters	6489	2545	10	26532	0.148	1.477	0.147	1.471
SIAM2007	21519	7077	22	500	0.101	2.226	0.089	1.952

Enron

One of the results of the Enron bankruptcy proceedings was the release of the emails of senior management of Enron to the public domain (Klimt & Yang, 2004) which has become an important corpus in NLP research. The emails themselves are organized in folders. Enron with categories is a filtered subset of the base Enron email corpus, which has been categorized by students of UC Berkley (UC Berkeley, n.d.). The experiments used the Enron with categories dataset, with the following modifications: emails with length over 500 words were split into multiple documents, with the same labels applied to each component as were on the source email. Additionally, similar categories with low representation were bucketed together to increase representation of the categories. Finally, defined training and testing sets were constructed through sampling the resulting dataset and applying thresholds on the number of emails from a given class. Word embeddings were pre-trained on the final dataset using the word2vec implementation in Gensim (Rehurek & Sojka, 2010).

SIAM2007

The SIAM 2007 dataset (Srivastava & Zane-Ulman, 2005), which itself is a subset of the publicly available Aviation Safety Reporting System (ASRS) dataset, was used for the SIAM 2007 Text Mining Competition. The dataset contains safety reports reported by various aviation personnel. A version of this dataset, TMC2007-500, is commonly used for classification experiments but lacks word sequencing as TMC2007-500 is available in a bag of words format. SIAM2007-500 is thus derived from both SIAM2007 and TMC2007-500 – taking the text from SIAM2007, and filtering for the reduced vocabulary of TMC2007-500, preserving word sequences. Word2vec word embeddings were pre-trained on SIAM2007-500 using Gensim.

Reuters

The Reuters 21578 (Lewis, n.d.) dataset is a staple of NLP research. The corpus contains categorized news articles from the 1987 Reuters newswire. The ApteMod (The Reuters-21578 benchmark corpus, ApteMod version, n.d.) variation of the Reuters dataset, is a reduced version of Reuters 21578, drawn from the financial newswire, and requiring that categories have at least one document in both the training and testing sets. As ApteMod is skewed (Williams) (as is Reuters 21578), the Reuters dataset used for the experiments considers only documents in the ApteMod variation which correspond to the 10 most common classes in the dataset. The corpus was sourced from NLTK (Bird, Loper, & Klein, 2009), tokenized, and mapped to publicly available word embeddings trained using FASTTEXT (Bojanowski, Grave, Joulin, & Mikolov, 2017; Joulin, Grave, Bojanowski, & Mikolov, 2016; Mikolov, Grave, Bojanowski, Puhrsch, & Joulin, 2018).

Amazon

The Amazon (He & McAuley, 2016) dataset used is a collection of product descriptions, alongside category information, ranging from software to music to electronics and beyond. The provided categories are grouped together into 8 new yet related categories. For example, the original categories of "Children's", "Children's Books", "Children's Music" and "Electronics for Kids" are grouped together to form a group reflecting children. Each category is provided a threshold, and a subset of item descriptions are randomly sampled until each category has satisfactory products. Train and test sets are then created from the resulting data, with 300-dimension word embeddings pre-trained using Gensim.

Experiments

The experiments entailed training and evaluating performance of each of the implemented DNN models on the previously listed datasets, three times each – leading to three sets of results per dataset per DNN model. Each experiment consisted of 800 epochs of training on the training set, with model evaluation on the test set each epoch. A fixed number of epochs was chosen with runtime in mind. As such, the reported results are not necessarily the best performance of the model on a given dataset, but the best performance attained within 800 epochs of training. DNN model performance is evaluated against each other within this constraint.

Evaluation Measures

Multi-label classification introduces complexity into the evaluation of accuracy, due to class skew and sparseness (Han, Kamber, & Pei, 2012), and fundamentally, what

is a prediction which only partially matches the correct labels considered as – a hit, a miss, or something else.

One approach is exact match – where either all of the predicted classes match the expected labels, or they do not – in essence treating the outputs in a sort of binary fashion. This method is particularly harsh as a classifier can have some proficiency in identifying a subset of classes and still be useful.

Outside of an exact match, we can consider each output separately, evaluate their predictive power individually, and averaging the results to get a representative measure of performance. Two approaches to this are macro and micro averaging (Manning, Raghavan, & Schutze, 2009). With Macro averaging, the measure value for all labels is simply averaged, treating all labels as equal. Given this, macro averaging does get impacted by class imbalance. The alternative is micro averaging, which is more resistant to class imbalance. Micro-averaging sums the truth-values for each label together (i.e. the true positives for all labels together, all the false negatives together, etc.), and then calculates the measure from these totals.

The macro averaged versions of the following measures are used in the result evaluation. In the listed formulae, *TP* is the number of True Positives, *FP* is the number of False Negatives, *FP* the number of False Positives, *N* the number of examples and *L* the number of labels.

To evaluate the accuracy of the models a number of evaluation measures are used to get a well-rounded view.

Precision

A measure reflecting the proportion of identified positive classes which are true (Perry, Keng, & Berry, 1955). A system with high precision will by definition have a low number of false positives but may be missing many potential matches – that is, yielding a large number of false negatives. Precision is calculated as:

$$P = \frac{TP}{TP+FP}.$$

Recall

A measure which indicates the proportion of correct label assignments that is returned by the classifier (Perry, Keng, & Berry, 1955). In many ways, recall is the complement to precision, as a system with high recall will correctly identify classes for the input but can also include many incorrect labels as well. It highlights the impact of false negatives and is calculated as:

$$R = \frac{TP}{TP+FN}.$$

F1

The harmonic mean of precision and recall (Van Rijsbergen, 1979; Yang & Liu, 1999), the F_1 score is a measure of the accuracy of a classifier. As both recall and precision have deficiencies (i.e. not considering false positives or false negatives respectively), F_1 provides a means of incorporating both these measures to mitigate the deficiencies. F_1 is also known as the Dice similarity coefficient and is calculated:

$$F_1 = \frac{2*P*R}{P+R}.$$

The related $F_\beta$measures which weight precision or recall heavier are not considered, as specific applications of the DNN are not being discussed with the focus being general performance.

The following are complementary measures which take true negatives into account. They do provide further information into the behaviour of the models.

Informedness

The probability of making an informed decision (Powers, 2007) versus chance, informedness is derived from recall and its mirror. As the probability of a false prediction increases informedness will decrease, and is calculated:

$$I = \frac{TP}{TP+FN} + \frac{TN}{TN+FP} - 1.$$

Markedness

What informedness is to recall, markedness is to precision. Calculated from precision and the mirror of precision, markedness is a measure of how many predictions are correct (Powers, 2007).

$$M = \frac{TP}{TP+FP} + \frac{TN}{TN+FN} - 1.$$

MCC

The geometric mean of informedness and markedness, the Matthews correlation coefficient (MCC) (Matthews, 1975) is another measure of accuracy and balances the influence of TP, FP, TN, and FN (Chicco, 2017), reducing bias.

$$MCC = \sqrt{I * M}$$

A different approach to accuracy evaluation is through loss functions. While loss functions are used during the training of the DNN to gauge error and adjust accordingly, they can also provide a lens into the model's operation afterwards.

Hamming Loss

The ratio of false predictions to the total available labels, Hamming loss is a measure of the error of the classifier (Sokolova & Lapalme, 2009). Being a loss function, the closer to 0 the value, the better the performance indicated.

$$H = \frac{\sum_{i=1}^{N}(FP_i + FN_i)}{NL}.$$

Threshold

The values returned by the output layer of the DNN can be viewed through the lens of probability, and be considered to be the probability that label l_i applies to the input with probability o_i, the value returned by output i. Using this, a threshold t_i is applied to o_i, with o_i being considered as predicting l_i for the input, if $o_i > t_i$. Different approaches can be taken to determine t_i – choosing a constant value for all t_i (say 0.5), applying linear algebra (Zhang & Zhou, 2006; Elisseeff & Weston, 2002), or reviewing ROC (Han, Kamber, & Pei, 2012) and Precision/Recall (Davis & Goadrich, 2006) curves. For the purposes of this paper, the F1 score is calculated at every relevant threshold, for each epoch, label and result file. Of the returned scores for each epoch, label, result file combination, the maximum F1 and corresponding threshold is identified. The maximum F1 score is taken as it will yield the maximum MacroF1 and is a traditional measure of multi-label classifier performance. The same threshold is used for all other measure calculations.

Chapter 4. Results

Each model was evaluated three times with each dataset. The results of the experiments are summarized in Tables 2 and 3 below, showing the best result per experiment in the format of: median (minimum, maximum).

It is observed that model performance is relatively consistent for all the models on the Amazon and Reuters datasets (Table 2). More concretely, while there is some slight variation in observed performance metrics across the models, QRCNN consistently performs with top results, PSRCNAA and PSRCNN perform similarly to each other (and are next in line to QRCNN in performance), with HPSRCNN and RCLNN performing worst overall. Excepting Macro Recall, RCLNN performs quite a bit poorer than

Table 2

Median (minimum, maximum) best performance of macro-averaged result on Amazon and Reuters datasets; Bold identifies best performance of measure for dataset (by median); Measures are derived at the epoch from the corresponding F1

Macro Measure	**HPSRCNN**	**PSRCNNA**	**PSRCNN**	**QRCNN**	**RCLNN**
Amazon					
F1	0.747 (0.742,0.779)	0.894 (0.888,0.900)	0.893 (0.891,0.897)	**0.916 (0.914,0.921)**	0.510 (0.509,0.515)
Precision	0.760 (0.749,0.774)	0.906 (0.888,0.907)	0.916 (0.904,0.918)	**0.932 (0.926,0.935)**	0.409 (0.401,0.419)
Recall	0.745 (0.739,0.788)	0.890 (0.885,0.894)	0.881 (0.867,0.883)	**0.907 (0.900,0.908)**	0.783 (0.753,0.796)
MCC	0.681 (0.674,0.719)	0.865 (0.859,0.873)	0.864 (0.862,0.869)	**0.893 (0.891,0.899)**	0.375 (0.374,0.380)
Markedness	0.710 (0.680,0.716)	0.875 (0.875,0.878)	0.881 (0.872,0.884)	**0.903 (0.899,0.909)**	0.327 (0.322,0.327)
Informedness	0.668 (0.656,0.722)	0.857 (0.843,0.868)	0.856 (0.844,0.856)	**0.886 (0.881,0.889)**	0.446 (0.437,0.452)
Reuters					
F1	0.631 (0.621,0.722)	0.922 (0.922,0.923)	0.922 (0.882,0.924)	**0.929 (0.925,0.929)**	0.453 (0.429,0.469)
Precision	0.586 (0.562,0.703)	0.906 (0.896,0.909)	0.905 (0.881,0.908)	**0.907 (0.904,0.918)**	0.398 (0.350,0.400)
Recall	0.762 (0.757,0.765)	0.942 (0.940,0.950)	0.941 (0.885,0.942)	**0.949 (0.942,0.954)**	0.726 (0.702,0.763)
MCC	0.620 (0.610,0.703)	0.917 (0.916,0.917)	0.917 (0.873,0.918)	**0.923 (0.920,0.923)**	0.405 (0.397,0.423)
Markedness	0.565 (0.540,0.680)	0.896 (0.892,0.901)	0.895 (0.871,0.902)	**0.901 (0.899,0.912)**	0.337 (0.303,0.374)
Informedness	0.714 (0.708,0.736)	0.941 (0.935,0.942)	0.934 (0.878,0.940)	**0.941 (0.935,0.946)**	0.521 (0.501,0.557)

HPSRCNN. Both Amazon and Reuters have a label density around 0.5 and cardinality under 1.6 (Table 1).

With the Enron and Siam2007 datasets, performance is more varied (Table 3). There is no clear best performing model over all measures, which may be a result of the datasets themselves. The label density for these datasets is more extreme relative to Amazon and Reuters, being 0.089 and 0.402 respectively (Table 1). Label cardinality for both datasets is at least 1.95, which is larger than that of the previous datasets as well.

Table 3

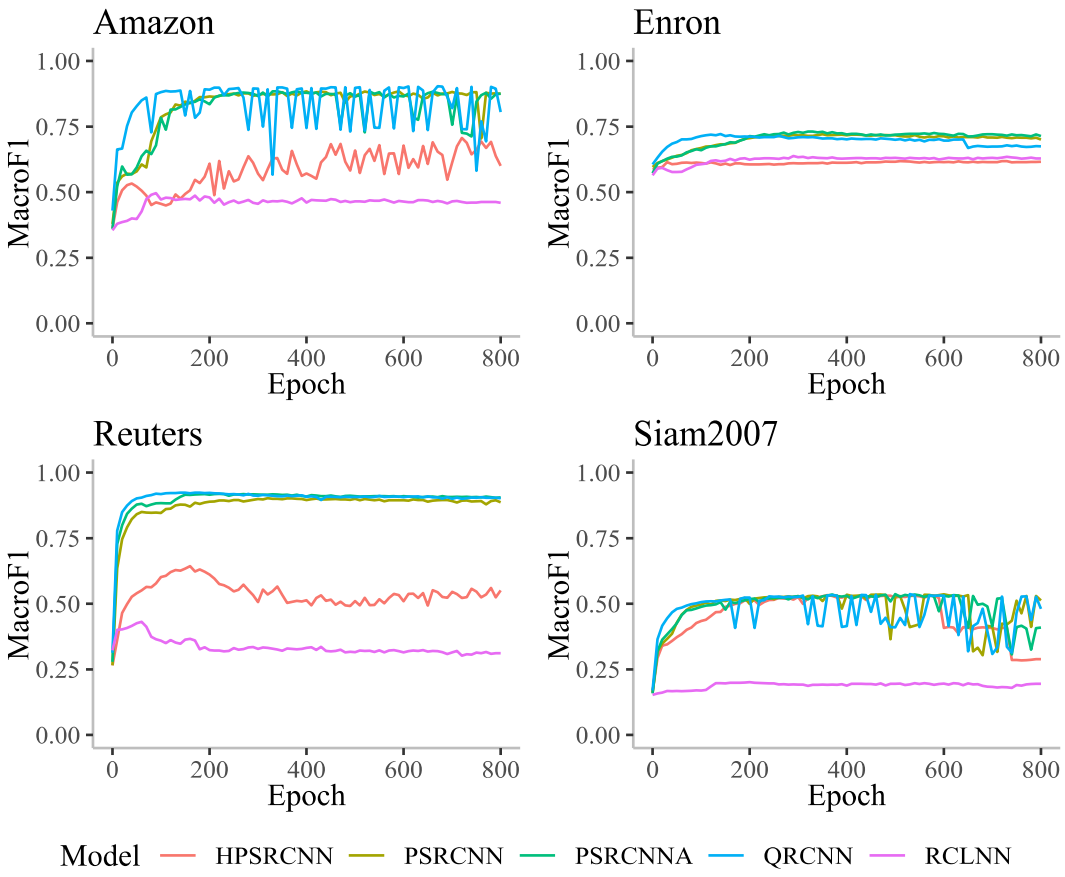
Median (minimum, maximum) best performance of macro-averaged result on Enron and Siam2007 datasets; Bold identifies best performance of measure for dataset (by median); Measures are derived at the epoch from the corresponding F1

Macro Measure	HPSRCNN	PSRCNNA	PSRCNN	QRCNN	RCLNN
Enron					
F1	0.623 (0.622,0.628)	**0.736 (0.735,0.737)**	0.732 (0.730,0.735)	0.729 (0.723,0.730)	0.639 (0.637,0.647)
Precision	0.505 (0.504,0.510)	**0.711 (0.704,0.720)**	0.698 (0.668,0.711)	0.643 (0.641,0.665)	0.541 (0.531,0.543)
Recall	0.903 (0.867,0.933)	0.800 (0.790,0.809)	0.813 (0.795,0.841)	0.844 (0.833,0.859)	**0.924 (0.877,0.937)**
MCC	0.312 (0.289,0.312)	0.551 (0.549,0.555)	**0.552 (0.549,0.552)**	0.538 (0.531,0.544)	0.378 (0.373,0.400)
Markedness	0.353 (0.294,0.375)	0.588 (0.580,0.625)	**0.596 (0.585,0.602)**	0.564 (0.555,0.578)	0.442 (0.408,0.456)
Informedness	0.281 (0.272,0.294)	**0.528 (0.492,0.528)**	0.518 (0.511,0.525)	0.515 (0.509,0.530)	0.358 (0.351,0.376)
Siam2007					
F1	0.540 (0.539,0.540)	**0.543 (0.541,0.543)**	0.542 (0.542,0.543)	0.542 (0.541,0.543)	0.222 (0.203,0.223)
Precision	0.574 (0.569,0.587)	0.579 (0.574,0.580)	0.578 (0.577,0.592)	**0.590 (0.583,0.592)**	0.159 (0.138,0.168)
Recall	0.526 (0.516,0.527)	0.527 (0.525,0.528)	0.523 (0.517,0.525)	0.522 (0.521,0.524)	**0.647 (0.513,0.658)**
MCC	0.491 (0.490,0.493)	0.494 (0.492,0.495)	0.493 (0.493,0.496)	**0.495 (0.494,0.496)**	0.137 (0.118,0.140)
Markedness	0.539 (0.537,0.543)	0.543 (0.535,0.544)	0.534 (0.534,0.548)	**0.548 (0.545,0.550)**	0.101 (0.087,0.104)
Informedness	0.452 (0.451,0.455)	0.456 (0.452,0.460)	**0.458 (0.453,0.460)**	0.456 (0.448,0.457)	0.209 (0.200,0.230)

Now let's consider some key measures. Macro F_1 provides an overall performance measurement, balancing precision and recall. The average macro F_1

Figure 8

Average Macro F_1



performance of the models over training time is highlighted in Figure 8. We can see that macro F_1 of QRCNN, PSRCNN and PSRCNNA tend to converge for each of the datasets and yield similar performance overall. Of the two pseudo-Siamese models, PSRCNNA generally outperforms and displays smoother convergence than PSRCNN, suggesting the addition of self-attention is having positive impact to the DNN. As well, PSRCNNA convergence is smoother than QRCNN. RCLNN has a generally flat macro F_1 curve, indicating little change is made via training. The macro F_1 curve for HPSRCNN presents an upward slope on Amazon and Reuters, suggesting that performance for HPSRCNN

could reach (or possibly exceed) that of the other models if more training epochs are provided.

It is observed that the macro F_1 curves for Amazon and Enron are performing quite differently, with the Enron curves particularly being smoother. Enron has a higher label density – so more examples of each label are exposed during both training and evaluation, leading to more consistent results. With that said, the increased label density can be contributing to lower performance as multiple labels may be conflated.

Figure 9

Average Macro Precision

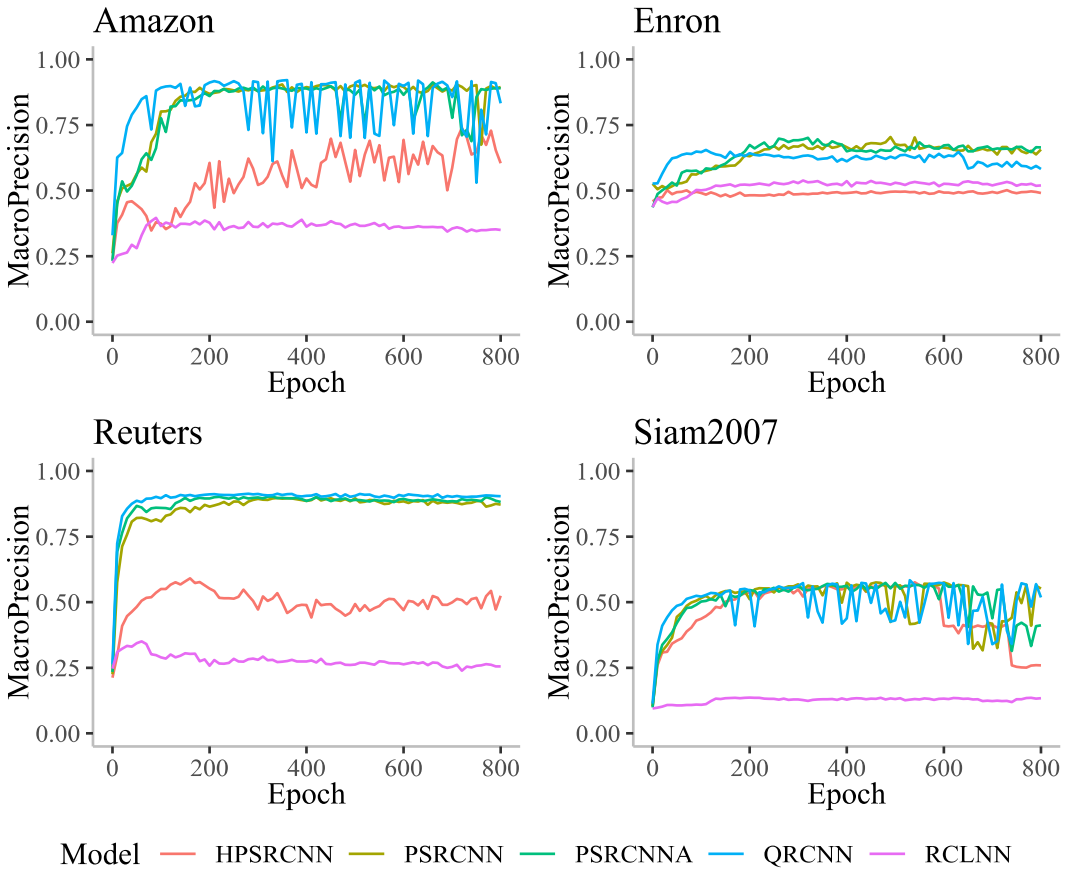
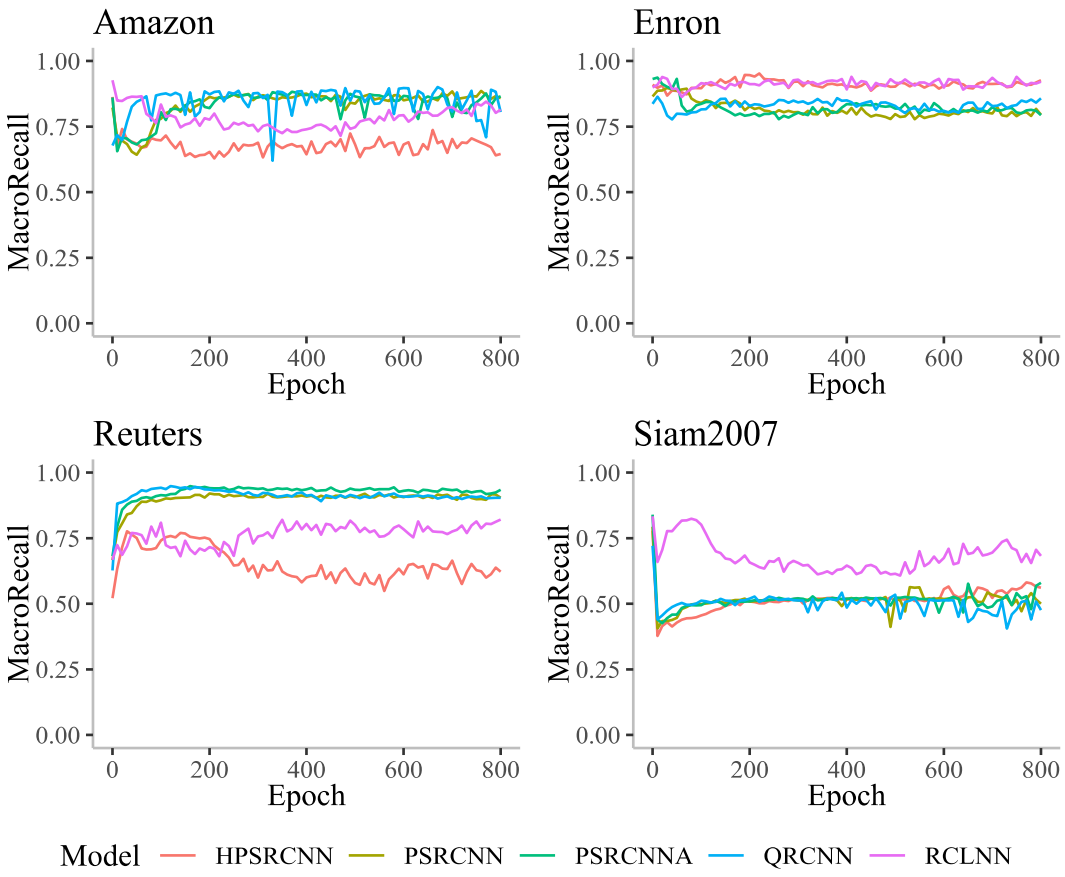


Figure 10

Average Macro Recall



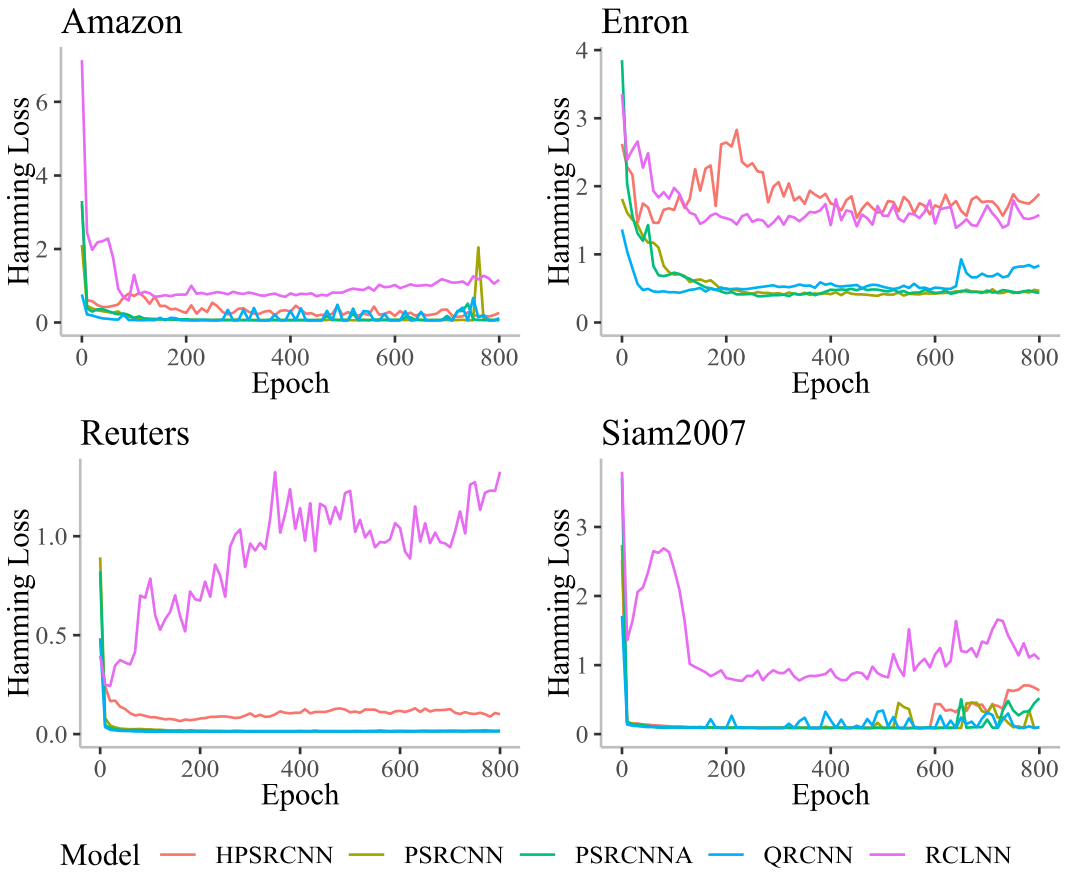
Precision and recall form the components of F_1. Precision is highlighted in Figure 9, recall in Figure 10. As we can see, the macro precision curve follows closely to the corresponding macro F_1 curve in Figure 8, whereas the macro recall curve does not, suggesting that the precision of the models is dominating the overall performance (as highlighted via macro F_1).

Another measure we can use to evaluate the performance of the models is Hamming loss, which provides a measure on the error of the models. From the Hamming loss curves (Figure 11) it is observed that RCLNN in particular, but also HPSRCNN have a tendency to diverge and grow the Hamming loss over training epochs,

whereas QRCNN, PSRCNN and PSRCNNA have tighter and less divergent curves – suggesting that both HPSRCNN and RCLNN have a higher error rate.

Figure 11

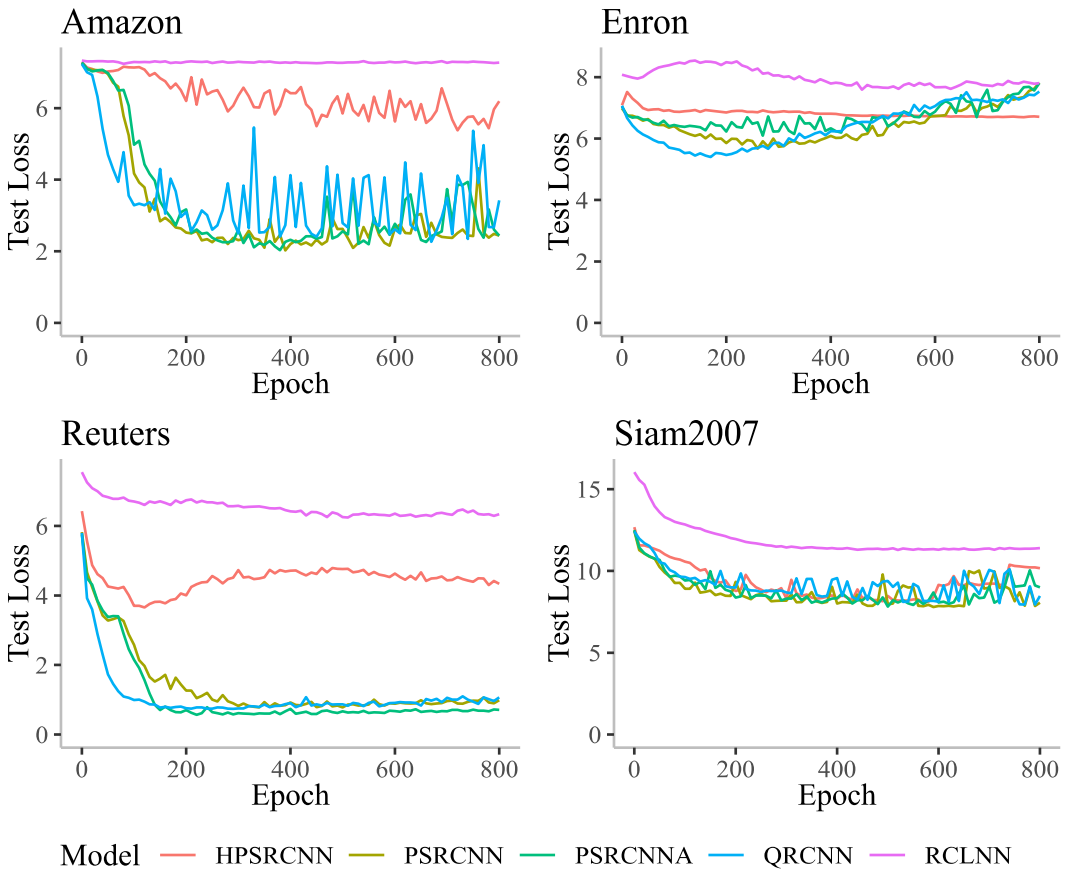
Average Hamming Loss



Another perspective into the performance of the models is presented in the train and test loss per epoch curves. Test loss (Figure 12) is calculated via the same loss function which drives the training and provides a more direct comparison of the models performance on the train and test datasets. One particular standout of the train loss progress (Figure 13) is how level it is for the models – the initial drop is quite quick and comparatively large, but then the curves flatten out. Test loss, by comparison, presents a much more saw-tooth shape generally, suggesting more inter-epoch variation.

Figure 12

Average Test Loss



The training time for models is another important comparator and is show in Table 4. While training time follows in part from the model itself, training time is also influenced by the dataset size (more data takes longer). Despite this, it provides insight into the operations of the model, and can influence model choice when combined with performance. QRCNN follows from Tables 2 and 3 as the generally best performing model, and Table 4 shows that it also has the second-best training time. The leader in training time, RCLNN, has demonstrated the poorest performance overall.

Figure 13

Average Train Loss

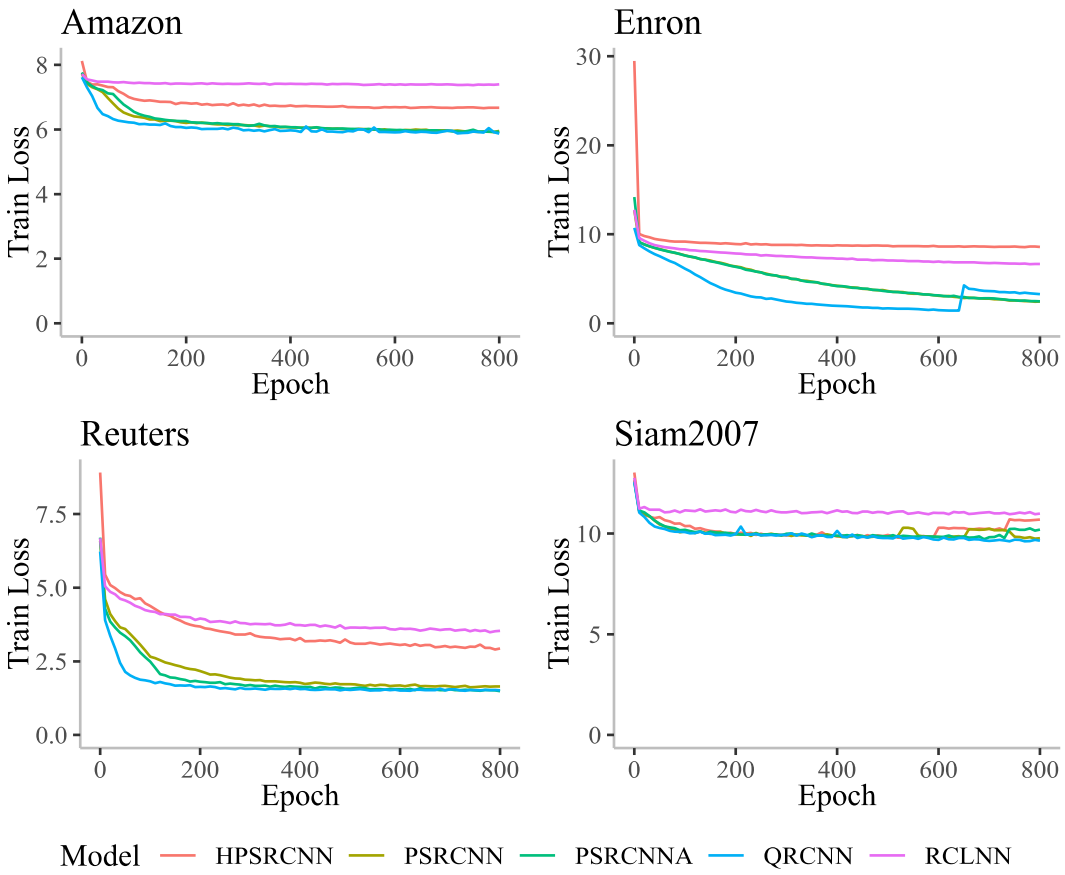


Table 4

Average runtime of one epoch (train + test) in seconds

Model	Amazon	Enron	Reuters	Siam2007
HPSRCNN	199	28	84	170
QRCNN	122	14	52	83
RCLNN	61	9	25	71
PSRCNN	162	21	68	141
PSRCNNA	181	23	75	154

Evaluation of each model in particular follows. RCLNN has general poor performance overall – the low values for informedness, markedness and *MCC* (Table 2 and 3) highlight this in particular, suggesting that RCLNN is prone to a large number of false predictions. RCLNN demonstrates its best performance (as measured by macro F_1) on Enron (Figure 8), which has the highest label density and cardinality. Macro F_1 performance of RCLNN increases as label density increases (Table 5). Figure 9 shows that on the datasets, RCLNN has stagnant or decreasing macro precision, whereas Figure 10 showcases RCLNN's increasing macro recall as training epoch increases. Looking at Hamming Loss (Figure 11), we see that RCLNN has a tendency towards divergence. Both Training and Test loss are fundamentally flat, showing a lack of convergence. The fastest time per epoch is observed with RCLNN (Table 4).

Table 5

Macro F1 performance alongside test dataset label density

Dataset	Density	HPSRCNN	PSRCNNA	PSRCNN	QRCNN	RCLNN
Amazon	0.200	0.747	0.894	0.893	0.916	0.510
Enron	0.402	0.623	0.736	0.732	0.729	0.639
Reuters	0.147	0.631	0.922	0.922	0.929	0.453
Siam2007	0.089	0.540	0.543	0.542	0.542	0.222

Conversely, QRCNN performs quite well on each dataset, and is top performer on Amazon and Reuters (Table 2), both of which have lower label cardinality (Table 1). Additionally the per-epoch time for QRCNN is quite short compared to the other models (second only to RCLNN)(Table 4). All of the measures in question are relatively close together. When a model outperforms QRCNN, it is not by a large margin (Table 3). QRCNN converges the fastest of the five models (Figure 13). The performance of QRCNN demonstrates the power of the context vector structure, integrating longer term

sequences, as simple max-pool to create document embeddings. The comparatively short per-epoch time married with the performance makes QRCNN the leading model of the five evaluated models.

Overall, PSRCNNA presents the best performance next to QRCNN by macro F_1 (Figure 8) – though when PSRCNNA outperforms QRCNN, the performance gap is generally small. It is observed that these instances correspond to the two datasets where the label density is at least two (so Siam2007 and Enron) (Table 5). As well, unlike when QRCNN is leading, PSRCNNA does not achieve peak performance in all supporting measures (Table 2 and 3). When we consider macro *MCC*, PSRCNNA is not nearly as performant (and QRCNN pulls ahead as the dominant model overall). The Hamming (Figure 11) and Test loss (Figure 12) curves are much smoother for PSRCNNA than QRCNN.

The performance of PSRCNN follows closely that of PSRCNNA (Table 2 and 3) - suggesting that the addition of self-attention to PSRCNNA was a meaningful addition. The one key observation pertains to the ranges of the evaluation metric values. When we consider F1, Precision and Recall, for PSRCNN the range is typically larger than the range for the corresponding measure of PSRCNNA (and when PSRCNNA has a larger range than PSRCNN, it is till typically tighter than the PSRCNN ranges). This observation does not hold for the *MCC* and related measures (Table 2 and 3).

Likewise, the HPSRCNN model does not attain top results – tending to be lower than those presented by the non-hierarchical models outside of RCLNN (Table 2 and 3). Similar to RCLNN, HPSRCNN presents rather poor performance on Enron, with the gap between F_1 and *MCC* (both macro and micro) suggesting that there is a large influence of

true negatives (Table 3). Training time is the longest of the five models considered (Table 4).

Chapter 5. Conclusion and Future Work

In this paper, the problem of multi-label classification of longer text documents was considered, through the lens of deep neural networks. Particularly, two key questions were highlighted – how can extended term dependencies be represented in learned representations and does the integration of learned local and global word context lead to better document representations. To address these questions, I have proposed five DNN models, two base models which are variations and extensions of existing DNN (QRCNN and RCLNN), and three hybrid approaches, integrating QRCNN and RCLNN. These new models were evaluated on four datasets with varying results. Fundamentally, RCLNN is not capable of producing informative results – at least within 800 epochs. RCLNN's poor performance could be sourced from a few possibilities. The internal convolutional component of the RCL may not be clearly identifying meaningful features. Another possibility is that the RCL itself may not be detecting meaningful features over longer sequences, unlike what has been observed in computer vision applications. Thirdly, there is a dimension reduction in RCLNN – implemented before the convolutional component of the RCL, and then reverted before leaving the RCL. This reduction in dimension may be discarding necessary information to support the feature discovery of the RCL. The poor performance of RCLNN likely has impact on the performance of models which incorporate it.

It appears that the hierarchical structure of HPSRCNN is polluting the embeddings – given the lower performance observed in relation to the non-hierarchical models – and HPSRCNN not identifying key salient features to aid in classification. The generally poor performance is likely rooted in a couple possibilities. The choice of an

arbitrary sentence length (10 words) could be a contribution to this, as well as the compounding of the use of RCLNN (which has demonstrated generally poor performance). The generally slow rate of convergence of HPSRCNN suggests, like RCLNN, that there is much more training capacity, and performance may very well improve given enough training time. However, the duration of one epoch would make training a challenge.

Performance of PSRCNNA and PSRCNN is undoubtedly influenced by the poor performance of RCLNN – but not terribly so, given the overall performance of these pseudo-Siamese. These models, as well as HPSRCNN include residual connections. While the intent behind these connections is to improve training, the connections are introducing uncontextualized information alongside the contextualized output of RCLNN and QRCNN – which may be weakening the learned representation (and hence lowering performance). The self-attention mechanism of PSRCNNA looks to provide some compensation for the influence of RCLNN and the residual connection, leading to better general performance than the non-attention variant. Both PSRCNN and PSRCNNA performance is close to QRCNN, so close in fact that it suggests that QRCNN dominates the operation of PSRCNN and PSRCNNA, with minimal influence of RCLNN. The alignment of the loss values supports this as well.

Following this, it would appear that the integration of embedding information between the models evaluated has no material impact. The longer-term dependencies in this instance were not reinforced by the local context brought by convolution, and RCL did not demonstrate capacity to identify or integrate extended dependencies – as highlighted by QRCNN dominating the PSRCNN and PSRCNNA performance. Given

QRCNNs leading performance, both overall and in conjunction with epoch duration – it appears to be sufficient to consider preceding and following global context, without a need to reinforce local context into embeddings.

Answers

Topics are distributed throughout a document, which requires consideration of the entire text to identify them. So how can we ensure that the learned representations of DNN fully incorporate these extended dependencies? In the context of this paper, this means better multi-label classification. From the experiments it is found that QRCNN is top performer overall, indicating that incorporating the global context of a word into that word's embedding is crucial. RCLNN, the other non-integrated model considered, performed substantially worse, suggesting the approach that QRCNN takes to integrating context identifies the subtopics (i.e. labels) better than the approach of RCLNN. PSRCNNA performs close to QRCNN, but with generally smoother convergence, highlighting how attention can compensate for deficiencies in the model.

To the second question – does a hybrid approach, integrating representations learned by different models provide better document classification results – the research of this paper indicates a resounding no. In particular, the non-hierarchical hybrid approaches PSRCNN and PSRCNNA had performance just shy of the key component model QRCNN. The hierarchical HPSRCNN performed substantially worse. From this, it would suggest that a single, focused model will produce a better representation than integrating the different perspectives of different models.

Future Work

The possible influencers of RCLNN's performance cannot be discounted and provides one path for future work. RCLNN performed rather poorly – quite possibly due to the factors identified previously. It is possible that eliminating the dimension reduction within the RCL would yield better performance of RCLNN and provide a better opportunity to evaluate the impact of integrating QRCNN and RCLNN embeddings. Alternatively, a different convolutional structure within the RCL could be requisite for better performance. The consideration of these opportunities would provide further insight into RCLNN's performance, and aid in determining if RCLNN (and by extension RCL) is applicable to the NLP domain.

The residual connections of PSRCNN, PSRCNNA and HPSRCNN may be weakening the embeddings through the inclusion of non-contextualized information. Further exploration of this idea – through the removal of the residual connections is a potential area for future work.

Regarding HPSRCNN, a good portion of its possible performance is tied to the performance of RCLNN. However, a different choice of sentence length (say dynamic as determined by punctuation) could provide a mechanism to improve performance. An alternative build of HPSRCNN (or PSRCNN, PSRCNNA) could also be considered, one which replaces the RCLNN component with a different recurrent convolutional neural network (or possibly purely convolutional). This would provide for further consideration of the integration of differently learned representations and continue with the idea of integrating local and global context. The specific mechanism on how the two

representations are integrated could also be reviewed – perhaps mean or max-pooling the two representations together instead of adding them as in the current models.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., & Devin, M. (2016). Tensorflow: A system for large-scale machine learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI).*

Allison, B., Guthrie, D., & Guthrie, L. (2006). Another look at the data sparsity problem. *International conference on text, speech and dialogue.* Berlin.

Bahdanau, D., Cho, K., & Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate. *arXiv preprint, arXiv:1409.0473*.

Balduzzi, D., & Ghifary, M. (2016). Strongly-typed recurrent neural networks. *ICML*.

Bellman, R. E. (2003). *Dynamic programming.* Courier Dover Publications.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends in machine learning, 2*, 1-127.

Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards AI. *Large-scale kernel machines, 34*(5), 1-41.

Bengio, Y., Frasconi, P., & Simard, P. (1993). The problem of learning long-term dependencies in recurrent networks. *IEEE International conference on neural networks.*

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is hard. *IEEE transactions on neural networks, 5*(2), 157-166.

Bernardini, F. C., da Sliva, R. B., Rodovalho, R. M., & Meza, E. B. (2014). Cardinality and density measures and their influence to multi-label learning methods. *Journal of the Brazilian Society on Computational Intelligence (SBIC)*, 53-71.

Bird, S., Loper, E., & Klein, E. (2009). *Natural Language Processing with Python.* O'Reilly Media Inc.

Bishop, C. M. (2006). *Pattern recognition and machine learning.* Springer.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of machine learning research, 3*, 993-1022.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the association for computational linguistics, 5*, 135-146.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., . . . Zieba, K. (2016). End to end learning for self-driving cars. *arXiv preprint, arXiv:1604.07316*.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. *Proceedings of COMPSTAT'2010.* Physica-Verlag HD.

Bradbury, J., Merity, S., Xiong, C., & Socher, R. (2017). Quasi-recurrent neural networks. *International Conference on Learning Representations.* Toulon.

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., . . . Shah, R. (1993). Signature ver- ification using a “siamese” time delay neural net- work. *International Journal of Pattern Recognition and Artificial Intelligence, 7*(4), 669-688.

Chen, C. P., & Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences,* 314-347.

Chen, G., Ye, D., Xing, Z., Chen, J., & Cambria, E. (2017). Ensemble application of convolutional and recurrent neural networks for multi-label text categorization.

2017 International Joint Conference on Neural Networks (IJCNN) (pp. 2377-2383). IEEE.

Chen, X.-W., & Lin, X. (2014). Big data deep learning: challenges and perspectives. *IEEE access, 2*, 514-525.

Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. *EMNLP.* Austin.

Chicco, D. (2017). Ten quick tips for machine learning in computational biology. *BioData mining, 10*(1), 35.

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint, arXiv:1409.1259*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint, arXiv:1406.1078*.

Chopra, S., Auli, M., & Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. *Proceedings of NAACL-HLT 2016.* San Diego.

Cohen, D., Ai, Q., & Croft, W. B. (2016). Adaptability of neural networks on varying granularity IR tasks. *arXiv preprint, arXiv:1606.07565*.

Das, A., Yenala, H., Chinnakotla, M., & Shrivastava, M. (2016). Together we stand: Siamese networks for similar question retrieval. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers.*

Das, R., Zaheer, M., & Dyer, C. (2015). Gaussian lda for topic models with word embeddings. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers).* Beijing.

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. *Proceedings of the 23rd international conference on machine learning.*

Dewey, M. (1876). *A classification and subject index for cataloguing and arranging the books and pamphlets of a library.* Amherst.

Dhongade, P., Longadge, R., & Kapgate, D. (2014). A review on classification of multi-label data in data mining. *International journal of computer science and mobile computing, 3*(12), 189-196.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *International workshop on multiple classifier systems.* Berlin.

Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., & Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. *Proceedings of the IEEE conference on computer vision and pattern recognition.* Boston: IEEE.

Du, C., & Huang, L. (2018). Text classification research with attention-based recurrent neural networks. *International journal of computers communications and control, 13*(1), 50-61.

Duchi, J., Hazan, E., & Singer, Y. (20111). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research, 12*, 2121-2159.

Elisseeff, A., & Weston, J. (2002). A kernel method for multi-labelled classification. *Advances in neural information processing systems*, 681-687.

Gargiulo, F., Silvestri, S., & Ciampi, M. (2018). Deep convolutional neural network for extreem multi-label text classification. *Proceedings of the 11th international joint conference on biomedical engineering systems and technologies (HEALTHINF 2018).*

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th international conference on artificial intelligence and statistics.* Sardinia.

Goodfellow, I., Bengio, Y., & Courville, A. (2016a). Convolutional Networks. In *Deep Learning* (pp. 321-361). Cambridge: MIT.

Goodfellow, I., Bengio, Y., & Courville, A. (2016b). Deep feedforward networks. In *Deep Learning* (pp. 163-220). Cambridge: MIT.

Goodfellow, I., Bengio, Y., & Courville, A. (2016c). Introduction. In *Deep Learning* (pp. 3-26). Cambridge: MIT.

Goodfellow, I., Bengio, Y., & Courville, A. (2016d). Machine learning basics. In *Deep Learning* (pp. 95-162). Cambridge: MIT.

Goodfellow, I., Bengio, Y., & Courville, A. (2016e). Optimization for training deep models. In *Deep Learning* (pp. 267-320). Cambridge: MIT.

Goodfellow, I., Bengio, Y., & Courville, A. (2016f). Regularization for deep learning. In *Deep Learning* (pp. 221-265). Cambridge: MIT.

Goodfellow, I., Bengio, Y., & Courville, A. (2016g). Sequence modeling: Recurrent and recursive nets. In *Deep Learning* (pp. 363-408). Cambridge: MIT.

Grnarova, P., Schmidt, F., Hyland, S. L., & Eickhoff, C. (2016). Neural document embeddings for intensive care patient mortality prediction. *arXiv preprint, arXiv:1612.00467*.

Han, J., Kamber, M., & Pei, J. (2012). Classification: Basic concepts. In *Data mining concepts and techniques* (pp. 327-392). Morgan Kaufmann.

He, K., Zhang, X., Ren, S., & Jian, S. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *Proceedings of the IEEE international conference on computer vision.* Las Condes.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings fo the IEEE conference on computer vision and pattern recognition.*

He, R., & McAuley, J. (2016). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *Proceedings of the 25th international conference on world wide web.*

Hilbert, M., & Lopez, P. (2011). The world's technological capacity to store, communicate, and compute information. *Science, 332*(6025), 60-65.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-R., Jaitly, N., . . . Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine, 29*, 82-97.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*, 359-366.

Hughes, M., Li, I., Kotoulas, S., & Suzumura, T. (2017). Medical text classification using convolutional neural networks. *Stud Health Technol Inform, 235*, 246-250.

IBM. (n.d.). *The four V's of Big Data*. (IBM) Retrieved 02 20, 2017, from http://www.ibmbigdatahub.com/sites/default/infographic_file/4-Vs-of-big-data.jpg

Jagannatha, A. N., & Yu, H. (2016). Bidirectional RNN for medical event detection in electronic health records. *Proceedings of the conference. Association for Computational Linguistics. North American Chapter. Meeting* (p. 473). NIH Public Access.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint, arXiv:1607.01759*.

Jozefowicz, R., Wojciech, Z., & Sutskever, I. (2015). An empirical exploration of recurrent network architectures. *International Conference on Machine Learning.*

Kalchbrenner, N., & Blunsom, P. (2013). Recurrent convolutional neural networks for discource compositionability. *arXiv preprint, arXiv:1306.3584*.

Kenter, T., Borisov, A., & De Rijke, M. (2016). Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint*.

Kim, Y. (2014). Convolutional neural networks for sentence classificaiton. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Doha.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *Proceedings of the 3rd international conference for learning representations.* San Diego.

Klimt, B., & Yang, Y. (2004). The enron corpus: A new dataset for email classification research. *European Conference on Machine Learning.*

Kong, X., Ng, M. K., & Zhou, Z.-H. (2013). Transductive multilabel learning via label set propagation. *IEEE Transactions on knowledge and data engineering, 25*(3), 704-719.

Koomsubha, T., & Vateekul, P. (2017). A character-level convolutional neural network with dynamic input length for Thai text categorization. *2017 9th International Conference on Knowledge and Smart Technology (KST).* Pattaya.

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems.*

Kurkova, V. (1992). Kolmogorov's theorem and multilayer neural networks. *Neural networks, 5*, 501-506.

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence.* Austin.

Le, Q. V., Ranzato, M. A., Monga, R., Devin, M., Chen, K., Corrado, G. S., . . . Ng, A. Y. (2011). Building high-level features using large scale unsupervised learning. *arXiv preprint, arXiv:1112.6209*.

LeCun, Y. (1989). *Generalization and network design strategies. Technical Report CRG-TR-89-4.* University of Toronto.

Lewis, D. D. (n.d.). *Reuters-21578*. Retrieved 04 20, 2019, from https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection

Liang, M., & Hu, X. (2015). Recurrent convolutional neural network for object recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* Boston: IEEE.

Liang, M., Hu, X., & Zhang, B. (2015). Convolutional neural networks with intra-layer recurrent connections for scene labeling. *Advances in Neural Information Processing Systems*, (pp. 937-945). Montreal.

Lipton, Z. C., Elkan, C., & Naryanaswamy, B. (2015). Optimal thresholding of classifiers to maximize F1 measure. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.*

Liu, J., Chang, W.-C., Wu, Y., & Yang, Y. (2017). Deep learning for extreme multi-label text classification. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.* Tokyo.

Liu, L., Liu, K., Cong, Z., Zhao, J., Ji, Y., & He, J. (2018). Long length document classification by local convolutional feature aggregation. *Algorithms, 11*(109).

Liu, P., Qiu, X., & Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. *Proceedings of the twenty-fifth international joint conference on artificial intelligence (IJCAI-16).*

Liu, Y., Liu, Z., Chua, T.-S., & Sun, M. (2015). Topical word embeddings. *Proceedings of the twenty-ninth AAAI conference on artificial intelligence.* Austin.

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint, arXiv:1508.04025*.

Lyu, C., Chen, B., Ren, Y., & Ji, D. (2017). Long short-term memory RNN for biomedical named entity recognition. *BMC Bioinformatics, 18*(462).

Manning, C. D., Raghavan, P., & Schutze, H. (2009). *Introduction to information retrieval.* Cambridge University Press.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure, 405*(2), 442-451.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics, 5*(4), 115-133.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint, arXiv:1301.3781*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2018). Advances in pre-training distributed word representations. *Proceedings of the international conference on language resources and evaluation (LREC 2018).*

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 3111-3119.

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data, 2*(1).

Neculoiu, P., Maarten, V., & Rotaru, M. (2016). Learning text similarity with Siamese recurrent networks. *Proceedings of the 1st Workshop on Representation Learning for NLP*.

Nowlan, S. J., & Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation, 4*(4), 473-493.

Parikh, A. P., Tackstrom, O., Das, D., & Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint, arXiv:1606.01933*.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *International conference on machine learning.*

Perrone, M. P., & Cooper, L. N. (1992). *When networks disagree: Ensemble methods for hybrid neural networks.* Brown University Inst. for Brain and Neural Systems.

Perry, J. W., Keng, A., & Berry, M. M. (1955). Machine literature searching X. Machine language; factors underlying its design and development. *American Documentation, 6*(4), 242.

Powers, D. M. (2007). *Evaluation: From precision, recall and F-factor to ROC, informedness, markedness and correlation Technical report SIE-07-001*. Adelaide: Flinders University.

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint, arXiv:1710.05941*.

Rehurek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks.* Valletta.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint, arXiv:1609.04747*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation.* No. ICS-8506. California Univ San Diego La Jolla Inst for Cognitive Science.

Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint, arXiv:1509.00685*.

Russell, S. J., & Norvig, P. (2010). Learning from examples. In *Artificial intelligence: A modern approach* (pp. 693-767). Prentice Hall.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing, 45*(11), 2673-2681.

Severyn, A., & Moschitti, A. (2015). Twitter sentiment analysis with deep convolutional neural networks. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval.* Santiago.

Shin, J., Kim, Y., Yoon, S., & Jung, K. (2018). Contextual-CNN: A novel architecture capturing unified meaning for sentence classification. *2018 IEEE International conference on big data and smart computing.* Shanghai.

Siegelmann, H. T., & Sontag, E. D. (1995). On the computational power of neural nets. *Journal of computer and system sciences, 50*(1), 132-150.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing and management, 45*, 427-437.

Spoerer, C. J., McClure, P., & Kriegeskorte, N. (2017). Recurrent convolutional neural networks: A better model of biological object recognition. *Frontiers in Psychology, 8*, 1551.

Srivastava, A., & Zane-Ulman, B. (2005). Discovering recurring anomalies in text reports regarding complex space systems. *IEEE Aerospace Conference.*

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks for overfitting. *Journal of machine learning research, 15*, 1929-1958.

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. *arXiv preprint, arXiv:1505.00387*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 3104-3112.

The Reuters-21578 benchmark corpus, ApteMod version. (n.d.). Retrieved from http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning, 4*(2).

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2009). Mining multi-label data. In *Data mining and knowledge discovery handbook* (pp. 667-685). Bostan, MA: Springer.

UC Berkeley. (n.d.). *UC Berkeley Enron email analysis*. (UC Berkeley) Retrieved 04 20, 2019, from http://bailando.sims.berkeley.edu/enron_email.html

Ushio, T., Shi, H., Endo, M., Yamagami, K., & Horii, N. (2016). Recurrent convolutional neural networks for structured speech act tagging. *IEEE Spoken Language Technology Workshop (SLT).* San Diego: IEEE.

Van Rijsbergen, C. J. (1979). *Information Retrieval.* London: Butterworths.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 5998-6008.

Vu, N. T., Adel, H., Gupta, P., & Schutze, H. (2016). Combining recurrent and convolutional neural networks for relation classification. *arXiv preprint, arXiv:1605.07333*.

Wang, J., & Hu, X. (2017). Gated recurrent convolutional neural network for OCR. *Advances in Neural Information Processing Systems.* Long Beach.

Wang, X., Jiang, W., & Luo, Z. (2016). Combination of convolutional and recurrent neural network for sentiment analysis of short text. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers.* Osaka.

Wen, Y., Zhang, W., Luo, R., & Wang, J. (2016). Learning text representation using recurrent convolutional neural network with highway layers. *arXiv preprint, arXiv:1606.06905*.

Williams, K. (n.d.). NLTK Reuters Readme. NLTK.

Yang, Y. (2018). Convolutional neural networks with recurrent neural filters. *arXiv preprint*.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. *Proceedings of the 22nd annual international ACM SIGIR conference on research adn development in information retrieval.* Berkeley.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* San Diego.

Yin, W., Ebert, S., & Schutze, H. (2016). Attention-based convolutional neural network for machine comprehension. *arXiv preprint, rXiv:1602.04341*.

Yin, W., Kann, K., Yu, M., & Schutze, H. (2017). Comparative study of CNN and RNN for natural language processing. *arXiv preprint, arXiv:1702.01923*.

Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2018). Self-attention generative adversarial networks. *arXiv preprint, arXiv:1805.08318*.

Zhang, M.-L., & Zhou, Z.-H. (2006). Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on knowledge and data engineering, 18*(10), 1338-1351.

Zhang, X., Henao, R., Gan, Z., Li, Y., & Carin, L. (2018). Multi-label learning from medical plain text with convolutional residual models. *Machine learning for healthcare conference.* Palo Alto.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Caracter-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, 649-657.

Zhang, Y., & Wallace, B. C. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint, arXiv:1510.03820*.

Zhao, Z., & Wu, Y. (2016). Attention-Based Convolutional neural networks for sentence classification. *The 17th Annual Conference of the International Speech Communication Association*, 705-709.

Zhou, Q., Zhang, Z., & Wu, H. (2018). NLP at IEST 2018: BiLSTM-Attention and LSTM-Attention via soft voting in emotion classification. *Proceedings of the 9th*

workshop on computational approaches to subjectivity, sentiment and social media analysis. Brussels.

Zhou, Y.-T., & Chellappa, R. (1998). Computation of optical flow using a neural network. *IEEE International Conference on Neural Networks.*