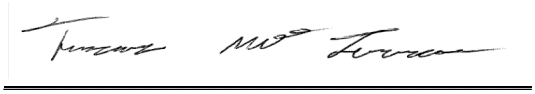
APPROVAL SHEET

Title of Thesis: Stratified Neural Models for Document Classification

Name of Candidate: Sarthak Mayur Mehta
Master in computer science, 2020

Thesis and Abstract Approved:



Professor Francis Ferraro
Department of Computer Science and
Electrical Engineering

Date Approved: March 2, 2020

ABSTRACT

Title of dissertation: Stratified Neural Models for Document Classification

Sarthak Mayur Mehta,
Master in Computer Science, 2020

Dissertation directed by: Professor Francis Ferraro
Department of Computer Science
and Electrical Engineering

Document classification is abstract task in the domain of natural language processing and information retrieval. There are traditional methods associated with this task, our method shows the performance enhancement in terms of the performance, convergence and enrichment of information.

We propose a hybrid neural language modeling architecture that constructs hierarchical feature representations. We examine our architecture through document classification. In our first model we begin with a character level convolutional neural layer (CNN) to get word level representation, next layers recurrent neural network (RNN) with attention based feature merging in order to get sentence level representation and again we have RNN with attention layer to get document level representation and finally, we have interconnected dense structure stacked to classify documents with soft-max activation. We extend this model to the word level and summarize the overall results and comparisons with baseline models.

We show evidence of the hypotheses on multiple datasets, utilizing IMDB

YELP review datasets. We show extended results with all datasets in terms of performance with F1 score, accuracy, precision and recall. Also, we show the comparison of convergence time and rate of convergence of our approach. Moreover, we show visual evidence that our approach lead to better feature construction and able to construct features for 99% of the effective word vocabulary from the characters in the documents.

Stratified Neural Models for Document Classification

by

Sarthak Mayur Mehta

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County in partial fulfillment
of the requirements for the degree of
Masters in Computer Science
2020

Advisory Committee:
Dr Frank Ferraro
Dr Tim Finin
Dr Tim Oates

Acknowledgments ‘

I would like to thank first and foremost, my advisor Dr Frank Ferraro for guiding with the direction for this contribution. I appreciate the insights, suggestion and advises provided from my advisor. Most importantly, I respect the patience, efforts and faith Dr Ferraro put in me and contributed his time and energy towards this thesis research.

I would like to thank lead of the retrieval group at National Institute of Standards and Technology (NIST) Dr Ian Soboroff for taking time from his schedule and providing views over this contribution. Also, I would like to thank Dr Tim Finin and Dr Tim Oates for taking part in my defence committee. I appreciate every individual input provided to accomplish this work.

I would also like to thank Mehdi Rezaee Taghiabadi, PhD candidate at ebiquity lab at Univerity of Maryland, Baltimore County (UMBC) for helping in proofreading. I thank High-Performance Compute Facility (HPCF) at University of Maryland Baltimore County for the providing the compute resources such as TAKI for this contribution.

Lastly, I would like to thank my friends and family for constant support and belief. Above all, I would like to thank the great almighty GOD for providing me strength and blessings.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

OCR	Optical Character Recognition
DF	Document Frequency
IG	Information Gain
MI	Mutual Information
CHI	χ^2-test
TS	Term Strength
KNN	K-nearest neighbor
SVM	Support Vector Machine
LSI	Latent Semantic Indexing
$TFIDF$	Term Frequency Inverse Document Frequency
CNN	Convolution Neural Network
RNN	Recurrent Neural Networks
HAN	Hierarchical Attention Network
GRU	Gated Recurrent Units
NLM	Natural Language Modeling
$LSTM$	Long Short-Term Memory
NMT	Neural Machine Translation
$RBMT$	Rule Based Machine Translation
SMT	Statistical Machine Translation
$BLEU$	Bilingual Evaluation Understudy Score
$CRCAN$	Character-Level Recurrent Convectional Attention Network
$WRCAN$	Word-Level Recurrent Convolutional Attention Network
$WGRU-ATT$	Word-level Gated Recurrent Unit with Attention function
$WGRU-MAX$	Word-level Gated Recurrent Unit with Maxpool function
$WGRU-AVG$	Word-level Gated Recurrent Unit with Average function
$CGRU-ATT$	Character-level Gated Recurrent Unit with Attention function
$BERT-AN$	Word-level BERT with Attention network

Chapter 1: Introduction

1.1 Philosophy in Language and Text by human representation

The language and text are the means of the communication and information representation used by humans in making interactions. The information representation not only conveys the addressed object and the subject for the particular event but also the context in which the object exists which related to the subject for that particular event and events intertwines and which exists in-universe, the protocol we use to do so is called language and transcript medium is called text. The singular unit to represent the part of the information in language is called lexicons or in other words, syntactic representation (example - the vocabulary of the person or dictionary, branch of knowledge, or type of language based on the region) with which most of the time the object is associated and the part of the reference information which is in context is called lexemes or in other words, semantic information (example – the meaning of sentence or inference) which is mostly related to the subject of the information.

When we represent any information, we use this set of lexicons and lexemes. This is where the ambiguity kicks in, one can have its unique way of representing this information with a unique set of the lexicons and lexemes which varies based

on the region, a branch of knowledge, age and context etc. When this information is exchanged in the medium, we observe that there exists a certain loss which is unavoidable. Example – for a part of the text "Pat and chandler agreed upon a plan. He said Pat would try the same tactic again". Here we see that we have ambiguity in the reference to "He". One can tell that from the given text, here "He" is referred to chandler, but it could be some other third person we do not know about based on the text. One can only tell if the context is given a prior, as similar to ambiguity. Thus, we can only understand the exact meaning if we are aware of the full context. This problem remains when there is an exchange of information representation between a machine and humans are involved, which is a domain of study within natural language processing known as natural language understanding which stems to the task of text classification. other related tasks are sentiment classification, part of speech tagging, relevance classification, learning to rank which again stems from text classification.

1.2 Document Classification as Task

Document Classification is the task of categorizing the documents according to their semantic and syntactic meanings, is an abstract task which stems from text classification. This task is a fundamental problem not only in natural language processing but also in other relevant fields like information retrieval [2]. The document classification has two subtasks, first is the feature construction and second is to classify those feature vectors into the respective categories. The feature construc-

tion involves the methods have syntactic and semantic signals getting superimposed. The semantic meanings can be extracted by finding the correlation between the sequence of words occurring together. Further details are extracted, by sequences of sentences appearing together, on the other hand, the syntactic meanings can be extracted by character level sequences. The finer abstraction of the semantic meaning can be further extended with merging higher level of stratification features which goes analogous to the given document. Thus, the hierarchical structure for document representations is valid candidates to extract a higher level of information from documents.

Machine learning in history has been proven to show the promising result in the domain of natural language processing tasks and it has been used extensively till now. The traditional heuristic-based approach showed some confidence in the researcher to employ the machine learning mechanism for the tasks seems NP-hard such as classification of the tweets into the categories of the mood of a person for understanding the crowd influence, classification of the product details and information into a particular set of the reviews, classification of the word or sequence of words to the next occurrence of the word, classification of the sequence of the words or segment of text for the task of translation, etc. These are some of the wells know use cases for the task of document classification. Other tasks in the domain of natural language processing are language -neural machine translation, a document summarizing, information entailment, part of speech tagging, grammar detection and sentence understanding, language modelling etc. In the domain of information retrieval text or language-based well-known tasks are recommendation

system, question-answer system, the task of learning to rank, natural language understanding and inference, etc. Even in extreme domains such audio or computer vision, there are applications associated with language such as phonetic processing of mood, speech modulation and translation, character identification in number plates, OCR scanning of the document, etc.

In next section, we will discuss the requirements for the implementation of the approaches of the task of natural language. We discuss the ideas and implementation in history in support for these tasks. We discuss the drawback and limitations that exists and hinders the performance enhancement in the field of the document classification. We present our hypothesis which are based on the observations from history. We show the detail explanation of our approach which supports the hypothesis, and finally we show the detail comparative study of the baseline models and our scripted approach. We show evidence of the implementation and experimentation, which reinforce the results and the comparison. Finally, we conclude the article with the summarization of the performance, the takeaway from the benefits of incorporating of our approach and some insights for the further work.

1.3 Motivation and Hypothesis for the Approach

The traditional approaches construct features based on counts of vocabulary words [33], which is also known as Bag of Word feature construction method, one hot encoding of the word vocabulary and also TFIDF scores [28] for documents. These traditional approaches have been showing successful results, but they fail to

gather semantic meaning of the document. Recently, with the advancements in the neural models and discovering the collaborative filtering method, embedding lookup tables have been shown to be effective extraction of complex features from sequence of the character and word representations. Based upon that in next chapters, we show that we can construct finer features from utilizing the stratification of the document construction for the semantic meaning representation and merging with the syntactic meaning. We show the classification performance comparison with detail feature extraction and coarser feature extraction.

To extract the higher-level information from documents, we split the data into the stratification which is analogous to the document. We start with first splitting a document into the sentences, which are further tokenized into words and finally into characters. To represent these characters, we use an embedding lookup representation which adds a dimension to the data model. This complex data structure is hard to utilize by traditional approaches, even the recent models cannot handle this bulky data to get the information from the very bottom level. The lower level features are learned with the exposure of the different units of the document is observed and higher-level feature mapping is learned from the relation between the coarser features. Thus, we need a novel an approach that utilizes the coarser features to form the higher-level feature abstraction which boosts the task of the classification and regression.

In recent techniques, it has been observed that higher-level features are learned using the complex neural layers such as recurrent neural networks. They have been proven to perform outstanding results over the natural language tasks. However,

the recurrent neural models require a high volume of data to train with, and it takes a significant duration for the process of the training that high volume data. We need a faster mechanism which can extract hidden semantic meaning from the inter-related lexemes representation. The convolution layers a different type of deep neural layers which extract spatial characteristics from the input. There is a limited set of the information is extracted from each kernel however, they are significantly faster to train and require relatively lesser data to train. The convolution layers can train faster as the kernel size is fixed which is generally smaller than the input data. Thus, we need an approach which can construct the mapping of the finer features but relatively it can train faster with a lower volume of the training data.

The feature construction involves learning of the mapping of the coarser feature to the higher abstraction of finer features. The coarser feature is the lexical level representation of the documents and the further the mapping of features is learned with the interrelation between each stratification of the document. The document's hidden representations can lie in any level of the stratification of the document, which makes it crucial to utilize each stratification to compose the mapping of the finer features. The stratification of the document is distinctly based on the individual documents thus, we need a modular approach which can compose the features from this specific document structures. Our hypothesis is based on the above-mentioned observations for the devising an approach which is capable to overcome all the limitation and immune enough to generalize the task of classification.

Hypothesis 1: We utilize the stratification in the document to cast feature from one step of the representation to the next step. Eventually, we stack all the mod-

ules for the constructions of the final abstraction of the features. Thus, the model exhibits context aware feature construction utilizing the hierarchical inheritance of the relations.

Hypothesis 2: The convolution layers to cast features from the word-level representations obtained from the character-level, leads to better word representations and consequently extracting the syntactic meaning and then learning the semantic meaning with the recurrent neural layers, between words and sentences results in better classification rates.

Hypothesis 3: Using the hybrid structure of the neural layer in the architecture which includes utilizing the convolution neural layers and recurrent neural layers. The feature casting method takes the convoluted output and transforms the dimension which is feasible with the recurrent layer. This makes the convergence rate of the neural parameter learning faster than using an individual type of layers throughout the structure.

Hypothesis 4: We get a broader set of the abstract features using the feature casting approach from the coarser level representation. The coarser level feature is limited and has a manageable size in terms of the dimension for finer feature computation example there could be at least 26 characters at the coarser level to represent any words in the document. Thus, a wide set of rich feature improves the task of classification in terms of performance.

In the next section, we discuss the contributions for this domain by the researchers over the timeline and brief each mechanism. Then we summarize the overall takeaways from the approach and synthesize the community view for the

development of a robust approach of the task. We see the comparative insights and patterns from history in terms of contributions, and we choose the state-of-the-art baseline for further comparison of our approach in experimentation.

1.4 Our Solution in Brief

By the observation and hypothesis, we provide an approach which can tackle the limit witnessed in history. We provide a novel approach which is an extension of the work by Yang et al., introduced a hierarchical method for the task of document classification. We know that task is not limited to the task of classification, we show the benefits that are enabled by our approach as in a generalized manner. The task extends for natural language inference, natural language understanding, language modelling, learning to rank and so on. We start with enabling the modularity in feature extraction by utilizing the stratification within the document using the approach of the Yang et al. [45]. They show that the stratified features with the task of classification extract a deeper meaning representation of the document. The next part, we include is the use of convolution neural layers in a novel way proposed by Kim et al. [16]. They show a feature construction method utilizing different convolution kernel essentially extracts kernel specific characteristics from the document. We merge this extracted feature using convolution layer with recurrent neural layers with stacked with attention mechanism on top for variable time max-pooling. Finally, we apply the multi-class classification task on topmost extracted features, and we use the soft-max mechanism to compute the score in the probability

distribution. We examine our approach over having character level and word level representation as initial index terms. The reason is that those representations are lower in the stratification of the document which has more local syntactic meanings, which eventually leads to broader and deeper contextual meaning with the complex mapping of local lexicons. We apply Kim et al. approach to learn this mapping of coarser lexicon representation to the finer contextual representations.

Chapter 2: Related Work and Contribution Discussion

2.1 Primitive methods for the task

Feature extraction is the core task utilized for sentimental analysis, named entity extraction [19], natural language inference [3], natural language understanding [34], document classification and many more. In the feature extraction process, we take the input signal which is essentially a sequence of the real numbers or integer values. In natural language processing, we construct the features with a sequence of text and apply categorization tactics. Core task has been implemented by means of the document frequency (DF), information gain (IG), mutual information (MI), χ^2-test (CHI), and term strength (TS). The comparative study of those approaches has been compared to by [42]. Further, they show that categorization is applied through the multi-variant regression model, nearest neighbour classification, Bayesian probabilistic approach, decision tree, artificial neural network, symbolic rule learning, and inductive learning algorithm.

They show the complete comparative study for the text categorization task by mean of the historic approaches and the methods feature construction. In-text categorization task, we have text for the topic or category in a document, we automatically classify those set of text into a category. This is a task with the traditional

approaches generates features with large size of the dimension, as we can multiple sets of unique combinations within the text. To reduce this curse of the dimensionality, thresholding is applied before the standard categorization method. They summarize that IG [18], MI [29] and CHI [29] performances has been evident as almost equivalent in terms of performance. The IG method with K-nearest neighbour (KNN) reduce the dimension size by 98% and boost up the performance of average precision which has been observed similar with DF method, but the important take away is that DF is simpler and lower in computation requirement. IG and CHI methods are not advised as higher computation requirements, and so is TF [43] as it shows 50% reduction in vocabulary.

Document Frequency is a method where we compute the frequency of the term appearing in the documents. We set a predefined threshold on these computed values to determine the valuable terms within the corpus and then we perform classification methods. The assumption here is that as the method involves computing frequency over the number of documents observed, gives us that which term are significant in the contributing towards document representation and the terms which are insignificant and below the threshold. This creates a bias in the classifier's performance. Also, we assume that each term in the complete corpus, follow the Zipf's law characteristics and are independent.

By applying this method of threshold we achieve a reduction in the dimensionality of the vocabulary for the main task. Also, its a simple approach and scales with very large corpora. We can apply the map-reduce form of architecture for parallel computation to make this process faster with large corpora. However, there are

limitations for the DF method, which includes fine-tuning of the threshold value, independent sense of term appearance although dependent upon the sentence formation, lose the contextual meaning as it fails to extract the semantic meaning and many more.

All of these methods are included in primitive approach along with TFIDF count based feature construction method by Cavnar et al. for N-grams of the document text, they introduce the bi-gram feature which is responsible to boost up the performance. The system involves the computation of the N-grams of the input documents and frequency for each, which is can be done fast and robust manner. Then, we compute the distance of the document for each category and select the category with the minimum value for classification. They perform the task on multiple setups, first they performed the experiment for the language classification task with this method and achieved 99.8% performance and second they performed for the task for classification of computer-oriented newsgroups, where they achieved performance of 80% classification error rate. Although overall the performance for the method did not do so well, there have been contributions for the improvement

The learning methods for these extracted features are examined by Yang et al., they have a wide set of learning algorithms including Support vector method and Neural Network approach. Support Vector method (SVM) give is us a clear margin which separates the two cluster categories the best. We represent the data in the vector space and we apply the support vector methods which lookup for the hyper-parameters which gives us the best separation, with reducing the hinge loss and maximize the margins between data points. They show the detailed comparison of

the SVM method with their implementation of KNN, Neural Network and Naive Bayesian.

Yang et al. re-evaluates the KNN approach as the results mentioned previously [14] were not satisfactory high enough with other contributions mentioned in [41]. They perform the task of the classification with using KNN approach, where they convert the text or documents in the feature vector space and we get the majority of category by computing distances of those features and sorting them in order. We put a threshold over the distance and take the majority of the K nearest distances. There many distance metric computation for identification of the similarity such as cosine similarity, Mahalanobis distance, manhattan distance, quadratic or Euclidean distance, etc. A thresholding method was applied by Yang et al. in their implementation approach which results in boosting in their performance, they suspect that and provide evidence through their implementation that not using a thresholding method resulted into lower performance for Joachims approach.

In a neural network, the approach has been extensively employed for the task of classification, the main benefits of this approach is that it can work with wide dimensions of the features. The implementation and result comparison of this approach has been observed by Wiener et al. and Ng et al.. In the method proposed by Wiener et al., they show the performance analysis of the neural network method along with the extension previously proposed by Latent Semantic Indexing (LSI) method and also shows equivalent performance with the method of term frequency.

Before we go in the neural networks, it's important that we define logistic regression. Logistic regression is a method of classification of input data points or

samples with associated binary labels. The logistic regression provides the confidence value for the input sample to be categorised as a positive or negative sample. The idea is that it computes a line which segments the positive data points from the negative, by empirically computing slope and intercept of the line. This method has two approaches for the computing the line, first is by the method of gradient descent and second is by the method of linear algebra. In the method of gradient descent, we start with a manifold space defined through the loss function. We start at a random point in the manifold space and follow the slope direction based on the task of improvement in the loss function. If the task is to find the lowest parametric values for the cost function then we descent towards the slope, and if the task is to maximize the cost function then we ascent in the manifold space. Thus, we can define a cost function f_{loss} for the task of logistic regression as equation 2.1

$$f_{loss} = -p * log(p) + (1 - p) * log(1 - p) \tag{2.1}$$

Vanilla logistic regression can compute line for linearly separable data points and for computing the non-linearly separable data points we need the higher order features with logistic regression. In such situations, neural approaches are employed to compress the information content and only extract the feature which are significant and abstract for the task of classification. Further with the neural network, we apply feature selection method and thus we include extra regularization term in the loss function, so with equation 2.1 we get the equation 2.2

$$Loss = \sum_{i=0}^{i=n} f_{loss} + \lambda \sum_{i=0}^{i=n} R(\theta) \tag{2.2}$$

where $R(\theta)$ is the regularization function. Based on the characteristics of data set,

features from the data and feature construction method, regularization function is chosen applied over the θ weight parameters. There are much higher order regularization methods however majorly used methods are lasso and ridge regularization. With the lasso regularization, we take the L1 norm of the weight and also known for feature selection method. Here, L1 norm is the degree one error which is defined as Manhattan distance, which is sum of absolute difference in co-ordinate points. The loss function with lasso regularization after reducing the error term, the minimize it further forces the insignificant weight parameter to close to zero and only retains the features which are significant for the task of classification, thus it known for feature selection method. While in ridge regularization we take L2 norm of the weights parameters, and also known for constraint the parameter values between 0 and 1. Here, L2 norm is the degree two error also known as euclidean distance, which is sum of square of difference of coordinates [20]. The error loss reduces to the least and constraint the parameter values to grow abnormally higher, which avoids overfitting. The influence of the regularization factor is controlled with the Lagrange multipliers λ. We select the value of the Lagrange multiplier which is the hyperparameter to control the trad-off point of the bias and variance. Wiener et al. experimented complete analysis of the neural network approach, where they utilized two settings first are with the vanilla with input layer and output layer without hidden layer and second is the neural network with the hidden layer which is 3 layer structure with several neurons in hidden layer as a hyperparameter. The neural network takes the sequence of the input words and computes the non-linear mapping for the classification. Wiener et al. shows the interesting experiment which

is evaluating the combination of the multi-class classification and the binary individual category classification, however, they miss the evaluation comparison with multi-class classification combine with just one category binary classification. Yang et al. showed the complete multi-class category classification on Reuters 21578 with 90 different categories and also neural network per category. Also, they showed the neural network with k hidden layers where k is the hyper-parameter.

Further, Yang et al. also examines the Support Vector Method (SVM) formally introduced by Cortes and Vapnik, where the task remains the same for the classification but the objective changes by finding the separators or support vectors which separates the data points by most optimal fitting line between them, where logistic regression would give the line which separates the data point but might fail to give the best separation. The problem of support vector is shown as the following equation 2.3

$$\vec{\theta}.\vec{x}_i - \vec{b} = 0 \tag{2.3}$$

which is similar to the problem equation of the logistic regression, but we provide additional constraints as shown in equation 2.4 and 2.5 which are applied to compute the margins and results in the optimal line on reducing the loss function.

$$\vec{\theta}.\vec{x}_i - \vec{b} \geq +1, y = +1 \tag{2.4}$$

$$\vec{\theta}.\vec{x}_i - \vec{b} \leq -1, y = -1 \tag{2.5}$$

The solution for the SVM problem can be done by quadratic programming techniques. The cost function is sightly similar to the logistic regression which is shown as an equation. The error term in the loss function for the SVM method is in place

as hinge loss, applying constraints supplementary as mentioned in equation 2.4 and 2.5. We introduce a new Lagrange multiplier C which is similar and inversely proportional to the variable λ, which controls the effectiveness of the margins applied over the data point inseparable domain. On setting the value of parameter C higher leads to classification more sensitive to the margins whereas with the lesser value of C the classification has more bias. The hinge loss function is as following as equation 2.6 and 2.7.

$$f^i_{hinge} = max(0, 1 - \theta^T x_i), y = +1 \tag{2.6}$$

$$f^i_{hinge} = max(0, 1 + \theta^T x_i), y = -1 \tag{2.7}$$

The over all loss function for the SVM is with equation 2.8,

$$Loss = C \sum_{i=0}^{i=n} f_{hinge} + R(\theta) \tag{2.8}$$

We usually employ the L2 norm or ridge regularization for the cost function, but we can also put the L1 norm or other convex function for this term $R(\theta)$ applied over the weight parameters. The SVM gives the optimal solution because the distance between the data points is with exactly $\frac{1}{||\theta||^2}$ are taken into consideration for the computing the finding the lines [44].

In history, we have seen the utilization of the convolution neural network(CNN) in the domain of computer vision. The basic idea starts with extracting handcrafted features which are called SIFT descriptors [21], which are spatially scaled invariant of the input and finally we use a classification algorithm such as logistic regression, KNN or SVM. This has been extended to the idea of neural network for the feature compression and similarly to the domain of deep learning where the feature is

extracted and learned with the training [17]. Also, the concept of max-pooling is analogous for the compression activity in neural network [47][11]. Inspired by the computer vision domain, we have seen the contribution of the CNNs in the text classification by Sermanet et al. and Zhang and Wallace. Recently, for the deep feature construction, we have seen the utilization of the convolution neural networks (CNN) and recurrent neural networks (RNN). Associates at Facebook AI research showed a very deep neural architecture which is based on CNN's applied for the task of text classification [6], the model performs feature extraction over character embedding and they show that by increasing the depth of the model, the performance increases. The problem with this model is the lack of context-awareness and the hierarchy of the abstraction which our approach promises.

The multi-modal information representation based approach by Wang et al. combines the information extracted from word-level representation, knowledge base context representation and the character level representation, using CNN layers followed by a concatenation layer. LeCun et al. applied CNN layers for the text local features, since then we have seen the contribution of CNN in basic NLP tasks, such as semantic parsing, search query retrieval, sentence modelling and many more. Kim provided a method to apply parallel CNNs for the multi-feature extraction, by applying various types of filters to the sequence of words embedding vectors for the sentence classification. Further, they apply global max-pooling and concatenate the extracted features. The application of the CNNs on the character embedding was applied by Zhang et al. for the text classification, they showed that their method generates competitive results with word features based methods. In 2016, for the

task of language modelling Kim et al. presented a character-based method for neural language modelling (NLM).

In 2014, Kim [15] showed a very interesting approach of applying the CNNs in the domain of text classification. The approach is based on the extracting different characteristics of the words appearing together by applying convolutional kernels. The kernels are used to specifically casts the n-grams window-based features from the word representations. The word representations are pre-trained word embedding which are generated using an unsupervised approach by Mikolov et al. [22]. Kim philosophy is that the neural network can perform better with using the initial word embedding. They show an experiment to their hypothesis and approach for the task of the question classification and sentimental analysis. The novelty is with the feature construction approach which merges the multiple channels of information using time distributed max-pooling, the results they show are comparable with the recurrent neural approach but they were not significantly higher than that. As the window based approach is applied over the sequence of words, not usually the words are correlated with each other adjacently. Most of the words entail to the different phrase, which could be within the sentence or in outside the sentence closer. They fail to show the performance of the approach with the large size of the sentence. They fail to preserve the network structure of the semantic meaning of the text, and only extracts the spatially correlated features for the classification. On the other hand, we provide a larger set of evidence of the comparison of the CNN approach of Kim applied over the sentence and characters.

Network that preserves the semantic meaning within the text clustered within

a document and approach to extract features from it was presented by Yang et al., where they utilized the hierarchy of the document structure, named as Hierarchical Attention Network (HAN). They show the two important main benefits from this approach. First they utilize hierarchy of the document such as words forms the sentences and sentences form document, similarly they provide the neural structure with same hierarchy. Secondly, their approach shows that, the words used in different setting may have different weight in contribution which we can extract from the hierarchical representation. Thus they focus on the extraction of the context world from the document as much as it can been see through. The context within the word and sentence highly depends on the setting and the entailment i.e it could be possible that same word or sentence could mean differently in different setting. This concept connects back to the main philosophy of the natural language representation of ambiguity and oblivious context.

To extract the information based upon the contribution, they utilized attention mechanism by Bahdanau et al. and also the inspiration of adopting the same mechanism in our approach. They perform 2 layers of attention mechanism for the sentence feature construction from word representation and the document layer features from the sentence representation. The attention mechanism provides selective weight to individual word to represent its contribution while sentence feature construction, such that we preserve the context meaning as the word used in a different setting would have different weights. Similarly, we repeat this process for the document representation from sentence persevering the context information from different sentences occurring in a different setting.

In our approach, we provide complete context preserving and also utilizing the syntactic meaning which merges with semantic meaning. We apply two layers of attention mechanism over character level time distributed max-pooling. Their approach is limited to the utilization of document structure on the word-level and they show experiments for merging the extracted latent features from RNNs. They employ the stacked Gated Recurrent Units (GRUs), followed by an attention mechanism proposed by Bahdanau et al. and Xu et al. for classification, which we follow similarly. The hierarchical approach was also employed for NLM task and sentence classification to show performance based on context-aware predictions, here Long Short-Term Memory (LSTM) is used for the recurrent calculations [10]. Zhou et al. [50] introduced LSTM, BI-LSTM and C-LSTM based methods for text classification, where C-LSTM methods involve the convolution-based feature extraction process and utilize the hierarchy of the document modelling. They show the hierarchy which is the sentence based hierarchy i.e within the sentence, phrases are linked for the contextual representation. However the contextual representation can be linked to different sentence, words within the sentence might not contextually be connected to the adjacent words in the phrase. They apply the convolution neural network-based filtering over the word to extract the semantic meaning from the words. They also show the comparative studies for the convolution filter design with the recurrent neural design (LSTM).

Recently, more hierarchical work was observed in form of sequence to sequence prediction for the task of neural machine translation (NMT) [52], they have a slightly different idea of hierarchy, they divide the long sentences into small sub-

text sequence which stems from Ghosh et al. and employ hierarchical structure for processing. First, the shorter sequences are processed by LSTMs and generate abstraction which is further processed with LSTM to have an abstraction of the overall text. With this task, we have a neural model which translates the one human language into another human language. There are three ways to do these translations, rule-based machine translation (RBMT), statistical machine translation (SMT) and neural machine translation (NMT). There have many approaches in history for the NMT and we the number of the NMT task have been increasing yearly. The progress in the state of the art method of the NMT can be tracked with a workshop of Machine Translation (WMT) [52]. The initial sequence to sequence work for NMT was observed by Sutskever et al. and Cho et al.. They have encoder-decoder structure, where the input to the network is the sequence of the word in one language and the decoder output is the sequence of words with another language with similar contextual information. The encoder translates the natural language into the ground machine language and the decoder translates back in the natural language from the machine language. Thus, we also call this structure as seq2seq networks, as has input sequence and reproduce the same type of sequence with different syntax by the same semantic meaning. They utilize a variant of the recurrent neural networks which is called LSTM proposed by Hochreiter and Schmidhuber, in 1997. Google utilized the idea of sequence to sequence model and came with intensive training and structure for neural machine translation [39]. They show the performance comparison with the state of the art methods using the Bilingual Evaluation Understudy Score [25] (BLEU) and perplexity.

Yogatama et al. [46] showed a novel way of LSTM trained, as a generative network in the presence of the discriminator. Their method outperforms other LSTM based approaches based on the text classification rates. Serban et al. [30] showed to apply hierarchical approach with the generative adversarial networks. They used the generative approach for the task of the movie dialogue generation and applied the encoder-decoder based hierarchical recurrent neural network. They

Contribution over graph-based approach for the sentiment classification is shown in Ebrahimi and Dou [9], where they use a chain based graph structure of RNN cells and similarly in Peng et al. [26], utilizing the CNN layers, this work stems from the contribution by Zhu et al. [51] and Socher et al. [32].

Character-word based text classification model by Jaech et al. [13], which utilizes two-layer of CNN layers, residual network and finally RNN layer for text classification, but their approach limits to extend the hierarchical approach. RNN-CNN hybrid structures based models which also utilize the hierarchical structure have been employed for the task of sentimental classification [36]. However, the same type of convolution filters are applied and learned overtraining phase, wherein our approach, we employ Kim et al. way of convolution, and we show that the results are promising.

Chapter 3: Models

In this chapter, we introduce our proposed models for text analysis and also show the base line models that we used for the comparison. As we are extending some of the baseline methods for the document classification task to character level inputs, we show the detailed study of the models. The baseline methods are the methods proposed by Yang et al. and then we extend their models with our approach to character level. Our main approach includes the convolution layers proposed by Kim et al. [16] to extract and merge the character features, followed by recurrent neural network layers similar to Yang et al. [45], then we add attention mechanism to construct the document features and finally a softmax layer is added to do the classification task. We show the complete implementation detail for the models, and show the performance of these models in terms of classification metrics and also the speed of convergence. For the building implementation of the machine learning models and approach, we propose for the development including convolution-based method proposed by for the word and sentence level feature construction, we start with character level and word level embedding. After performing the convolution processing [15], we push the representation into the recurrent neural network by for the forward and backward relations and finally, we merge them to form a document

representation [45] using attention mechanism [1]. Details for our models are as follows:

3.1 Character-Level Recurrent Convolution Attention Network (CRCAN)

In this model, we gather information from the characters. The first layer contains parallel CNNs which are used to extract the features from the character-level representation of a word [16]. These features extracted from different filters of CNN are concatenated to have word-level features of sentences.

Let V_c be the set of unique characters (vocabulary) and d_c be the embedding dimension. Then the embedding matrix E is of size $d_c \times |V_c|$. Each word w_k is a sequence of character indexes, defined as follows:

$$w_k = \left[c_1, c_2, \ldots, c_{l_k}\right], \quad 1 \leq c_i \leq |v_c| \tag{3.1}$$

We extract the relevant columns of E to represent each index as a vector in $\mathbb{R}^{d_c}$ and apply the padding to create $C_k \in \mathbb{R}^{l_c^* \times d_c}$, where l_c^* is the fixed character sequence length. The convolution is performed over C^k with filter $\mathbf{H}_c^{d \times w}$:

$$\mathbf{f}_c^k[i] = \tanh\Big(\big\langle C^k[*, i : i + w_c - 1], H_c\big\rangle + b\Big), \tag{3.2}$$

where w_c is the window size. We apply 2d Max-pooling over $\mathbf{f}_c^k$, defined as:

$$x_k = \max_i \mathbf{f}_c^k[i]. \tag{3.3}$$

For a sentence of length T_1, with the word features $x_k^j, k \in [0, T_1]$, we use Gated Recurrent Units (GRU) in both forward and backward directions to capture the

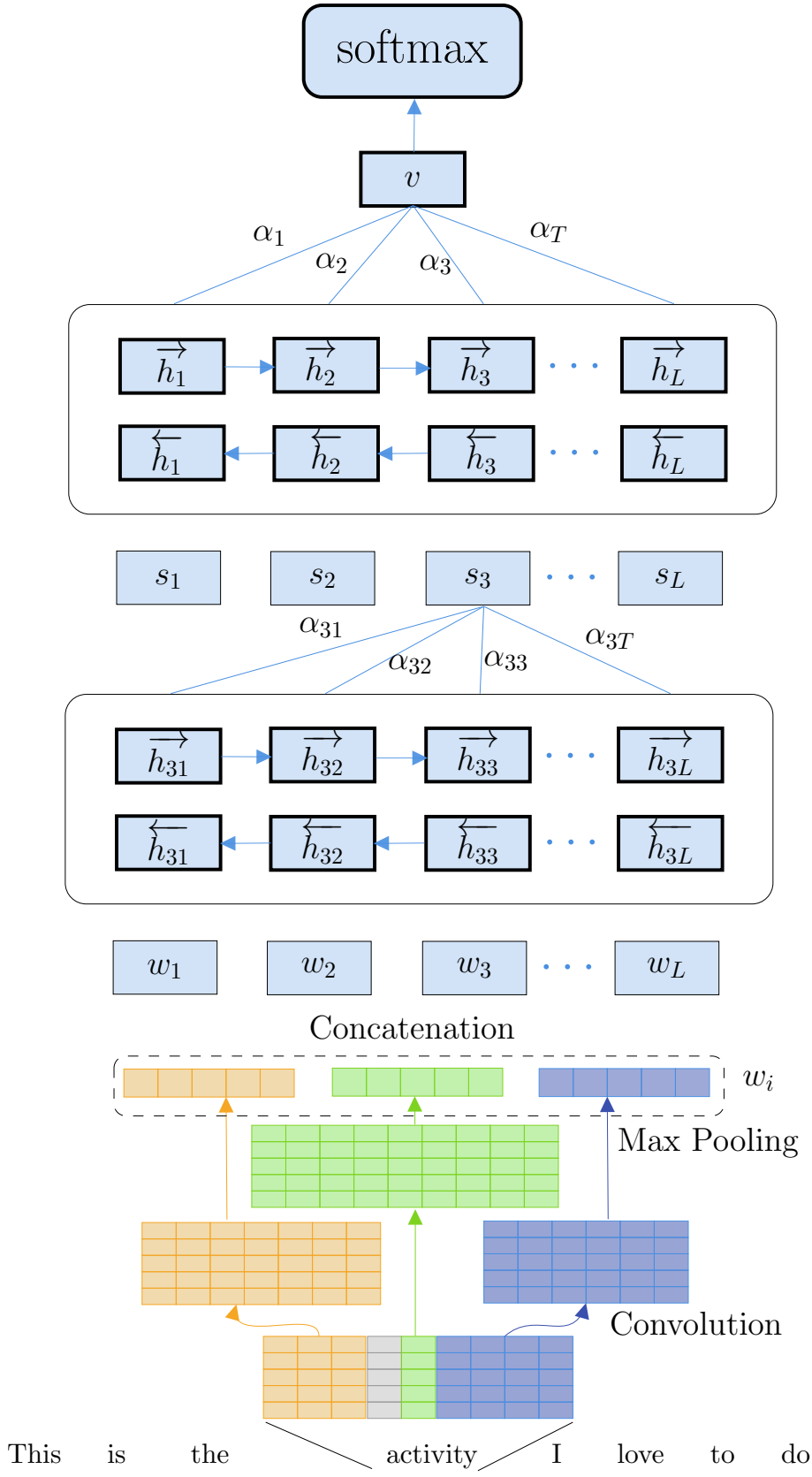


Figure 3.1: Character-level Convolution Recurrent Attention Network

sequential information. The $\overrightarrow{\text{GRU}}$ forward function computes the hidden representation $\overrightarrow{h_k^j}$ by extracting the semantic meaning between forward direction of sequence of words to represent the dependency of each word given previous words. Similarly $\overleftarrow{\text{GRU}}$ computes the backward sequence of word semantics $\overleftarrow{h_k^j}$ to gather the words information based on the next words. Concatenating these two representations provides semantic dependencies in both directions:

$$h_k^j = \left[\overrightarrow{h_k^j}, \overleftarrow{h_k^j}\right]. \tag{3.4}$$

Yang et al. show that not all the sentence representations around the word contribute equally for the sentence feature constructions. Thus, we utilize the weighted linear combination of hidden representations:

$$v^j = \sum_k \alpha_k^j h_k^j, \tag{3.5}$$

where

$$\alpha_k^j = \frac{\exp(u_k^{j^\top} u_w)}{\sum_t \exp(u_k^{j^\top} u_w)}, \tag{3.6}$$

and

$$u_k^j = \tanh(W_w h_k^j + b_w). \tag{3.7}$$

Similarly, we apply the same set of layers with GRU and the attention networks for the document feature construction.

3.2 Word-Level Recurrent Convolutional Attention Network (WRCAN)

In this model, in contrast with CRCAN, we utilize the word-level information. The overall structure is similar to CRCAN, but our aim is to extract the word-level feature representations of a sentence.

Let V_w be the set of unique words. Then the embedding matrix $Q_w \in \mathbb{R}^{d_w \times |V_w|}$ represents the words in the embedding space. For the sentence y_k we have:

$$y_k = \big[w_1, w_2, \ldots, w_{l_k}\big], \quad 1 \leq w_i \leq V_w \tag{3.8}$$

where each w_i is a word index and l_k is the number of tokens in y_k. Like CRCAN, we extract the embedding columns for w_1, w_2, ..., w_{l_k} and then do the padding to create $W_k \in \mathbb{R}^{l_w^* \times d_w}$, where l_w^* is the fixed word sequence length. We perform the convolution as follows:

$$\mathbf{f}_w^k[i] = \tanh\Big(\big\langle W_k[*, i : i + w - 1], H_w\big\rangle + b\Big), \tag{3.9}$$

where w is the window size. The next steps are similar to CRCAN as follows:

1. We concatenate the forward and backward GRU computed representations. (See Eq. 3.4)

2. We calculate the linear combination of hidden representations. (See Eq. 3.12)

3. The last layer is doing the softmax operation to classify input documents.

We next discuss the experimental results, the comparative performance for the different models, and the implications from the results.

3.3 Word-level Gated Recurrent Unit (WGRU-X)

In this model, we first start representing the document in the tensor with word index from the word vocabulary. Then, we apply series of GRU layers with a feature compression function X to cast upper stratified feature in the hierarchy. Finally, the task remain the same for the classification of the document by applying the soft-max function. In result section, we show the comparison of the different function applied for the feature compression.

We have given with word vocabulary V_w which is obtained from the document corpora with finding all the unique words from the training set. We cast a tensor for the input document, with dimension same as the document hierarchy and the series vector as the word indexes. We transform this series vector into a spatial representation, by applying a embedding table lookup l_w, each embedding vector is size d_w. This embedding lookup table can be loaded with pre-trained embedding vectors learned in word2vec fashion.

Then, we apply first layer of bidirectional GRU which extract the hidden information from the word sequence, in $\overrightarrow{h_k^j}$ and $\overleftarrow{h_k^j}$ as h_k^j, similar to equation 3.4. Then we apply the special function X for the feature to merge into the upper hierarchy representation. The special function includes three methods that we explore, first is average (AVG), max-pooling (MAX) and attention-mechanism (ATT) [1].

$$v^j = AVG(h_k^j) \tag{3.10}$$

$$v^j = MAX(h_k^j) \tag{3.11}$$

The equation of the function with average and max-pooling are as equation 3.10 and 3.10. Finally, we apply series of such stacked layers to form the document level representation and apply a soft-max based dense layer for the task of classification. In our case, we cast sentence feature from word and then document features from sentence features, thus applying two stacked layer of GRU with function X.

3.4 Character-level Gated Recurrent Unit (CGRU-X)

This neural model is similar to the word level model to WGRU-X, but in this model we extend with one more stratification i.e character level representation and apply feature construction methods. We represent the document with the series of the character indexes and cast the tensor which maintains the hierarchy of the document. These character indexes are from the character vocabulary which is constructed using the the unique characters within complete corpus.

Similar to the word level mode, we load the pre-trained character embedding lookup and transform the series of character index to the spatial representation of the dimension size d_c similar to equation 3.1, we also have option to learn those representation or fine tune the pre-trained embedding. Then we apply the three stacked GRU with the special function and finally we apply the softmax for the classification.

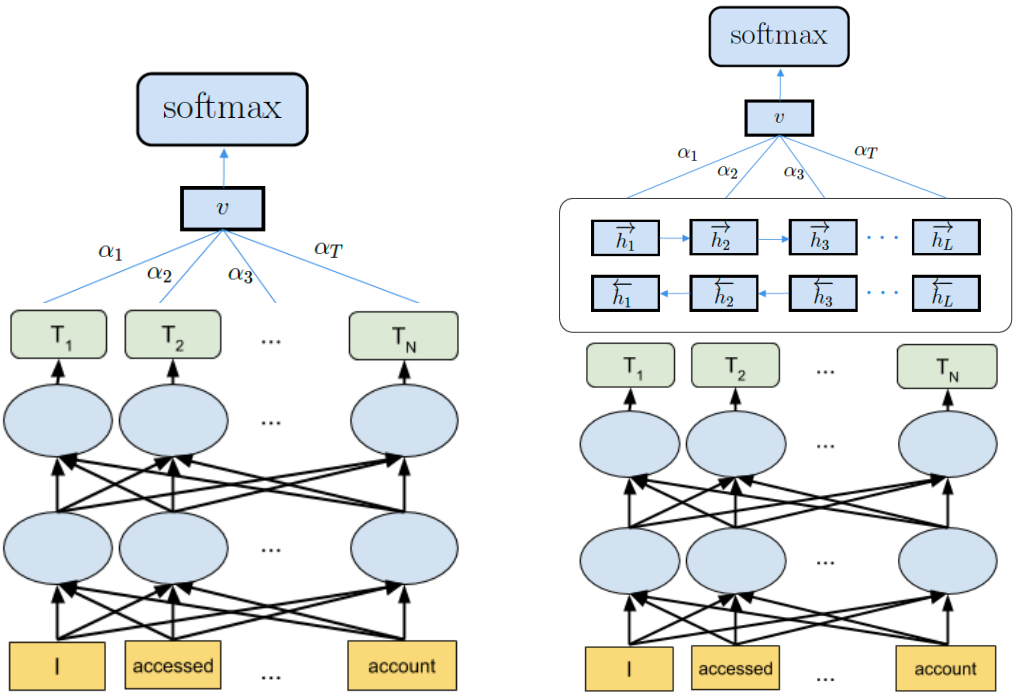


Figure 3.2: BERT Attention Networks: On the right side of the figure is BERT-AN(a) model and on the left side is BERT-AN(b)

3.5 Word-level BERT with Attention network (BERT-AN)

In the following work, we have seen the many implementation which load the pre-train embedding vectors such as Glove [22] and fastText [23]. We also examine the pre-trained encoder such as BERT - Bidirectional Encoder Representation from Transformers to examine the task of classification with documents. The BERT implementation is the stacked Bidirectional transformers which are loaded with the pre-trained representations, we attach the final layer for transformation and fine tune the top layer for the task. Thus, over all task takes two step, first is the loading the pre-train embedding and computing the transform and second is to

compute the fine tune weights for the specific task. The former step is similar to known ELMo feature based approach which is the work by Peters et al. [27], where the Bi-directional representations are added to the available model representation to improve the performance of the task. The latter step is based upon the fine tuning approach, such as Generative Pre-trained Transformer (OpenAI GPT) which enables to learn minimal task specific parameters over the downstream task and pre computes the transformation. Both the methods share the same objective to have un-supervised pre-trained representations and apply fine tuning on the upstream task, which is analogous to the BERT implementation.

The BERT implementation only enables the sequence of the word transformed into the sentence representation, however the hierarchical task of document classification [45] includes the multiple level feature construction. Thus we stack the time distributed BERT implementation over each sentence and apply the special function-X over the sentence feature extracted from the BERT to cast the document level feature. Finally, we apply soft-max for the task of classification. As the BERT provides the feature construction transformation word sequence and we learn the parameter if the we apply attention mechanism as special function and for the soft-max classification which is upstream task as shown in figure.

The BERT provides the vocabulary for word to index transformation V_w, we cast the tensor maintaining the hierarchy structure from the document. We pass each sentence S_j as a word sequence to the BERT transformers where the word w_i in sequence is index from the BERT vocabulary.

$$v^j = \sum_k \alpha_k^j S_k^j, \tag{3.12}$$

where

$$\alpha_k^j = \frac{\exp(u_k^{j^\top} u_w)}{\sum_t \exp(u_k^{j^\top} u_w)}, \tag{3.13}$$

and

$$u_k^j = \tanh(W_w S_k^j + b_w). \tag{3.14}$$

If we apply the attention mechanism as special function and we learn the weighted parameter from the sentence representations, which basically fails to extract the semantic meaning from the sentence relations. But it takes the weight contribution from each sentence so complete set of the sentence is taken into the consideration. Finally we apply the softmax algorithms as equation and we call this method as BERT-AN(a).

We also perform the analysis of the deeper neural model utilizing the BERT transformer using the bidirectional GRU for computing the sentence context-aware feature and we call it BERT-AN(b). The complete structure of both BERT-AN(a) and BERT-AN(b) is shown in fig. 3.2. As analogous to approach from Yang et al. [45], after the sentence feature construction from the BERT transformer, we pass through Bi-direction GRUs to extract the hidden information from both the direction of the sequence, similar to equation 3.4. Those contextual hidden representations are then passed through the attention mechanism to perform the time distributed aggregation (we call it special function X) and finally pass it to softmax

for the classification.

3.6 A brief digest of our architectures

Thus within this section, we show the implementation details and the mathematics for the mechanism of proposed methods. We showed the implementation design of the CRCAN method which is merging of the character features extracted from the convolution method and further we extract features using HAN method [45]. Similarly, we see the design for WRCAN method which stems HAN method and echos CRCAN method.

Also, we show the design for the baseline models which were proposed by Yang et al., we present the building blocks for those baseline models which are glued with the special function X. We define the special function and discuss the application in the next section. We also have extended baseline CGRU-X which is naive in expanding HAN models. The next section includes the evidence gathered after re-examining HAN models, which leads to the selection of Attention function (ATT) for baseline models results as CGRU-ATT.

Finally, we discuss the implementation of BERT method extension BERT-AN(a) and BERT-AN(b), specifically designed to modularize the architecture of BERT method for the task of document classification. In the result section, we show the implementation requirements for the experiments and discuss the comparative understanding of the models.

Chapter 4: Experimental Setup and Results

In this chapter, we show the implementation details, step to setup the experiments and discuss the comparative study of the results generated for the models. We performed the experiments over an high performance compute resource which has a graphical processing unit of 64 Giga-Bytes Nvidia V100, 16 Giga-Bytes of Free Memory Space and Intel Xeon(R) Gold 6140 CPU @ 2.30GHz processor. [1] All the experiments are performed in same setting for comparison in common base ground. We specifically load the document chunks into the memory and perform the stack of functions over it and log the performance over the decided metrics for the comparison. Thus, we stream the documents from disk to the memory having a parameter to control the batch size through our data pipeline.

4.1 Experimental Setup and Environment

We have a set of documents, which are completely transformed as index tensors. We extract the sentences from the documents and then tokenize the words based on the white space characters, and finally, we spilt those words into charac-

[1] Special thanks to UMBC computer systems resources taki: https://hpcf.umbc.edu/system-description-taki/

ters. The dataset is already been pre-processed such that all the words or tokens are normalized in terms of stop words and can be separated based on white space. We initialize word and character embedding space matrices, which are trained along with the other hyper parameters of the network. The number of characters in word is limited to 8, the maximum number of words in each sentence is 100 and the maximum number of sentences in each document is limited to 15 for each input tensor.

Dataset	**Classes**	**Average #s**	**Average #w**	**Average #c**	**Documents**
IMDB Reviews	10	14	325.6	1259.5	348,415
Yelp 2013	5	8.9	151.6	547.44	335,018
Yelp 2014	5	9.2	156.9	565.72	1,125,457

Table 4.1: Statistics of data sets: **#s** denotes the number of sentences , **#w** denotes the number of words and **#c** denotes the number of characters.

For our word level approaches, our experiments limits the size of vocabulary with a hyper parameter p, where p$\in$\{1000, 10000, 20000,..., length of word vocabulary\} tuned over the validation set performance. The remaining elements in the vocabulary are then replaced with $\langle$UNK$\rangle$ tokens, which is same for the character level approaches, but with relatively smaller length of vocabulary. In our experiments, since we have two kinds of vocabulary, we define $\langle \text{UNK} \rangle_W$ and $\langle \text{UNK} \rangle_C$ for unknown words and characters respectively. We load the pre-trained embedding lookup vectors to transform the sequence of the indexes into the vector space representation, which is then fine tuned over the training phase. For the word em-

beddings we use the GLove [22] and fastText [23] and for the character embedding lookup we use the fastText [23]. The filter design of the convolution contains the dimension of the size of the embedding vector and the window size varies. We have the embedding vector dimension size of 200, and we use 7 different kinds of filters having the window size $w \in \{1, 2, ..., 7\}$. We use a combination of different types of filters applied and merged together to extract the features. For the GRUs, we have bi-directions in series, and also our latent dimension is 50.

For the BERT implementation [8], we have predefined word vocabulary which has a mapping to the indexes. We load the documents to have indexes matching from the BERT vocabulary and cast the tensor. The identifier for the segmentation of sentence and padding values in the tensor is default set to all 0 and 1 respectively, as assumption includes that there will only one sentence taken into consideration and which has a full size of 100 words in sequence and no padding. BERT has 10 layers of stacked transformers and we have the flexibility to fine-tune.

4.2 Result Discussion

We perform our experimentations for the feature construction mechanism using the approach proposed by Kim et al. [16]. We evaluate the performance of the models based on the F1 score, recall, precision and accuracy over the validation and test sets on IMDB and YELP reviews. The data sets are derived into 80/10/10% for training, validation and test respectively. The statistics for the datasets are shown in table 4.1. First, we present the evidence for our main hypothesis with

supporting experiments, then we evaluate the training performance of the convolutional approach with the recurrent approach. Furthermore, we show that by using convolution we gain 99% of the possible vocabulary word feature reconstruction from the character features. Finally, we discuss some interesting benefits enabled by using our approach. To support our hypothesis, we present the experiments with a combination of the convolution layers with recurrent layers in different levels of stratification. There are two broad comparisons that we discuss, first is the comparison within character level stratification and second is the comparison with character level architecture with word-level architecture. Table 4.3, 4.4, & 4.5 are showing the comparison of the performance measure for the different models which include stratification from characters to document and words to document. These values are computed by taking the average of the 3 experimental runs for individual models. Following is the brief description of baseline models.

- **WGRU-MAX**: Word-based GRU layer followed by max pool function for the sentence feature construction, finally we apply same stacked layers for document feature construction [45].

- **WGRU-AVG**: Word-based GRU layer followed by average function for the sentence feature construction, finally we apply same stacked layers for document feature construction [45].

- **WGRU-ATT**: Word-based GRU layer followed by soft learning attention mechanism [1] for the sentence feature construction, finally we apply same stacked layers for document feature construction [45]. This model is mentioned

as HAN model.

- **CGRU-ATT**: Three sets of layers stacked together that contains GRU followed by soft learning attention mechanism. This model starts with the character embedding features learned over the training phase, passes through the stacked-layer to form document features.

In table 4.2, we show the comparison of the special function applied over the hierarchical feature construction. We observe that regardless of the data-set used for the task of document classifications, we see that attention methods outperformed the other two methods on all the metrics. As we re-evaluate the results which shows the matching performance with the comparative study was shown by Yang et al. [45]. Based upon the re-examination we decided to use the attention mechanism for the BERT method implementation extension and used as a baseline to compare with our approach of CRCAN. We show re-evaluation results over metrics of F1, precision, recall and accuracy, with three reduction functions AVG, MAX and ATT.

BERT-AN(a) and BERT-AN(b), is a variant of the BERT method based on hierarchical approach. In BERT-AN(a), we have 10 layers of BERT transformers used in time distributed format to convert word indexes into the sentence feature, from which last 3 layers we fine-tune over the training phase. Along with the fine-tuning of BERT, we also train the parameters for the attention mechanism and the output dense layer with soft-max activation for the classification. In BERT-AN(b), we keep the approach for the BERT transformer the same but we completely adopt the process of stratified learning. We keep the learning parameters unchanged for

BERT layer and only train the sentence level encoder along with adaptive weights of attention mechanism and the final output dense layer with soft-max activation for the classification. The soft-max based approach remains the same for the task of multi-class classification method.

Datasets	**Metric**	**WGRU-MAX**	**WGRU-AVG**	**WGRU-ATT**
IMDB	Accuracy	41.10%	39.44%	**44.53%**
	Recall	33.19%	33.60%	**39.94%**
	Precision	30.31%	30.90%	**39.32%**
	F1 Score	39.92%	40.48%	**47.37%**
YELP 13	Accuracy	63.90%	62.57%	**64.48%**
	Recall	30.11%	29.32%	**31.14%**
	Precision	30.24%	29.64%	**30.58%**
	F1 Score	**31.13%**	30.04%	30.37%
YELP 14	Accuracy	**66.75%**	66.65%	64.34%
	Recall	**32.04%**	31.61%	31.03%
	Precision	**32.13%**	31.82%	31.11%
	F1 Score	**32.27%**	32.25%	31.22%

Table 4.2: Document classification performance over test set, with measures of F1 score, Recall, Precision and Accuracy for the HAN methods

Metrics	Word - Level		Character - Level		BERT extensions	
	WGRU-ATT	**WRCAN**	**CGRU-ATT**	**CRCAN**	**BERT-AN(a)**	**BERT-AN(b)**
F1 Score	**47.37**	42.93	22.77	46.61	43.70	37.31
Precision	**39.32**	36.29	21.62	36.83	33.19	31.73
Recall	**39.94**	37.25	24.95	38.08	34.80	33.04
Accuracy	**44.53**	41.87	31.46	41.56	40.51	38.45

Table 4.3: The classification performance comparison of CRCAN and WRCAN with the WGRU-ATT - word level HAN model, BERT-AN(a) and BERT-AN(b) - extended BERT models and CGRU-ATT - character-level HAN models over **IMDB** dataset (test set) - Refer appendix A table A.1 and table A.2 for extended comparisons

Metrics	Word - Level		Character - Level		BERT extensions	
	WGRU-ATT	**WRCAN**	**CGRU-ATT**	**CRCAN**	**BERT-AN(a)**	**BERT-AN(b)**
F1 Score	30.37	31.16	26.65	**31.73**	30.40	30.68
Precision	30.58	30.58	26.62	**31.10**	30.09	29.80
Recall	**31.14**	30.39	26.71	30.75	29.99	29.52
Accuracy	64.48	63.77	57.74	**64.85**	63.12	63.51

Table 4.4: The classification performance comparison of CRCAN and WRCAN with the WGRU-ATT - word level HAN model, BERT-AN(a) and BERT-AN(b) - extended BERT models and CGRU-ATT - character-level HAN models over **YELP-13** dataset (test set) - Refer appendix A table A.3 and table A.4 for extended comparisons

Metrics	Word - Level		Character - Level		BERT extensions	
	WGRU-ATT	**WRCAN**	**CGRU-ATT**	**CRCAN**	**BERT-AN(a)**	**BERT-AN(b)**
F1 Score	31.22	32.29	27.99	**32.35**	31.52	32.06
Precision	31.11	30.58	26.73	**31.74**	30.83	31.59
Recall	31.03	**31.58**	26.15	31.44	30.37	31.38
Accuracy	64.34	**66.77**	58.30	66.35	64.95	66.50

Table 4.5: The classification performance comparison of CRCAN and WRCAN with the WGRU-ATT - word level HAN model, BERT-AN(a) and BERT-AN(b) - extended BERT models and CGRU-ATT - character-level HAN models over **YELP-14** dataset (test set) - Refer appendix A table A.5 and table A.6 for extended comparisons

4.2.1 Model evaluation and metrics comparision

CGRU-ATT is the character level naive extension of the HAN model, we utilize this model as a baseline to compare the performance of bi-directional GRU with parallel CNNs (CRCAN). The results show that within the character-level model, CRCAN outperforms CGRU-ATT with an overall improvement of average +8.8% for all metric measures across IMDB and YELP dataset. Further, comparing WRCAN and CRCAN with HAN shows that we have competitive performance for the IMDB dataset in 4.3. WRCAN and CRCAN outperform HAN over F1 score for YELP-13 and YELP-14.

We see that performance improvement on comparing the WGRU-ATT and WRCAN is not observed in the experiments in table 4.3, table 4.4 & table 4.5, as

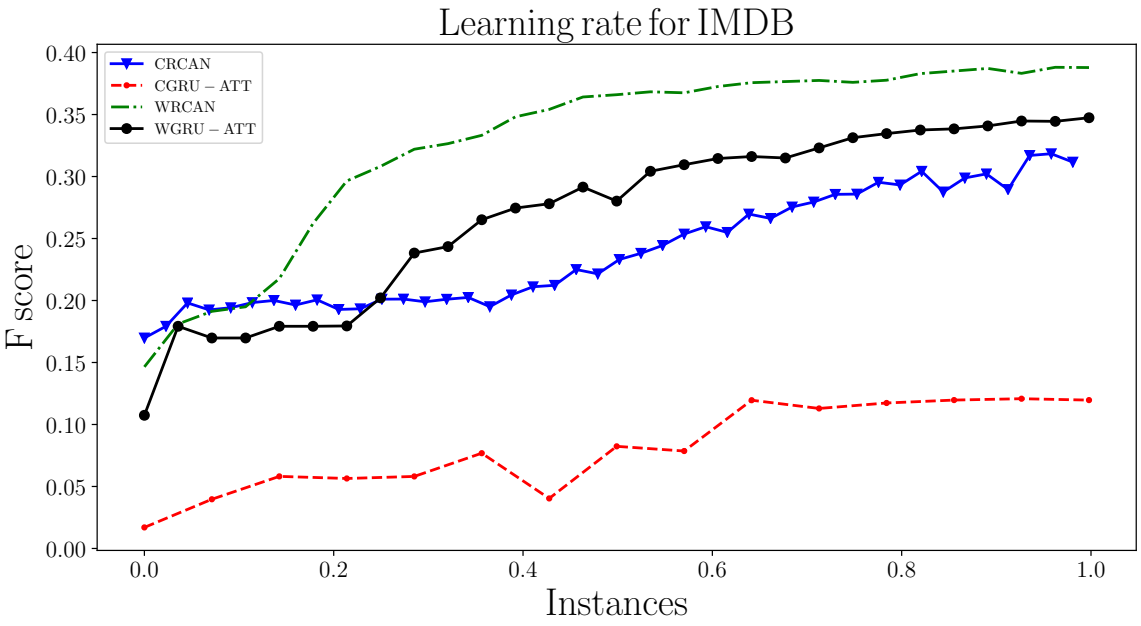


Figure 4.1: **IMDB** Comparison of the learning performance in terms of F1 score of character-level models which includes CRCAN and CGRU-ATT and word-level models which lincludes WRCAN and WGRU-ATT over an epoch.

Metrics	**WGRU-ATT**	**WRCAN**
F1 Score	10.64	17.26
Precision	1.21	1.5
Recall	10.01	9.98
Accuracy	6.4	6.6

Table 4.6: The classification performance comparison of our word level approach WRCAN with the word level HAN model over with SVM algorithm for **IMDB** dataset (test set)

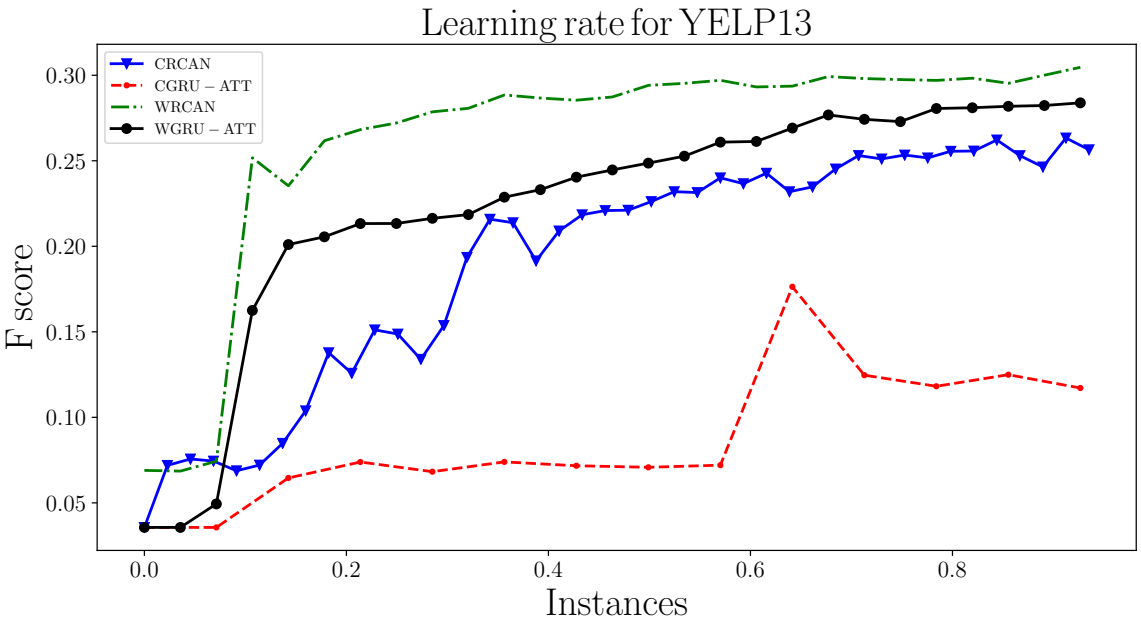


Figure 4.2: **YELP 13** Comparison of the learning performance in terms of F1 score of character-level models which includes CRCAN and CGRU-ATT and word-level models which lincludes WRCAN and WGRU-ATT over an epoch.

the kernel dimension of the convolution used for the information filtering process are much smaller than that of the input dimension space in word-level models i.e. there are relatively higher set of lexical information represented in the vocabulary and the kernel dimensions used for filtering are relatively much smaller. The filtering operation by the convolution is not efficient when we have a larger vocabulary and extremely smaller kernel, whereas it has been evident with the CRCAN models, that we observe significant performance improvement where the vocabulary dimension and the kernel dimensions of the convolution are equivalent in the dimensions. Thus, on applying the Kim et al. way of convolutions with the character level lexical representation have full contextual view of words in CRCAN results in word-level representation learned analogous to WGRU-ATT, in contrast, same method over

word-level lexical representation in WRCAN to learn sentence representation, loses the complete contextual view of sentence as words may have contextual information which entails out of binding of the kernel dimensions.

In table 4.3, 4.4 & 4.5, we also show the comparison of the BERT-AN implementation with the HAN models and the our approach. The performance of the BERT-AN(a) methods does not result in outperforming over all the metrics in comparison to the CRCAN. Similarly, BERT-AN(b) shows a competitive result with CRCAN on yelp-14 dataset, but the difference is not significant. We see that 99% of the word vocabulary from the datasets we are using matches the vocabulary of the BERT implementation. As the vanilla BERT is the task of the sentence classification, thus we present two variants of the BERT methods which utilize the stratified structure of the document for the task of classification. We see that BERT-AN(b) outperforms the BERT-AN(a) method, as former extracts not only the BERT sentence representation but also the context-dependent features which reinforce the claim of enhancing performance using the stratified contextual feature extraction. Later extracts the sentence feature and then we apply selective attention which fails to extract the sentence contextual features and results in lower performance. Thus, complete stratification is required in casting the document features for optimal contextual features along with syntactic features.

In table 4.6, we show the extended results over the different type of learning algorithm which is Support Vector Machine (SVM). It has a different philosophy for prediction than that of the soft-max algorithm. The performance optimization has not been observed with SVM as there is a heavy set of hyperparameters tuning is

required with more set of iterations. As the SVM algorithm is based upon optimizing the boundary separating the data clusters, which requires more training period for a heavy set of hyper-parameters.

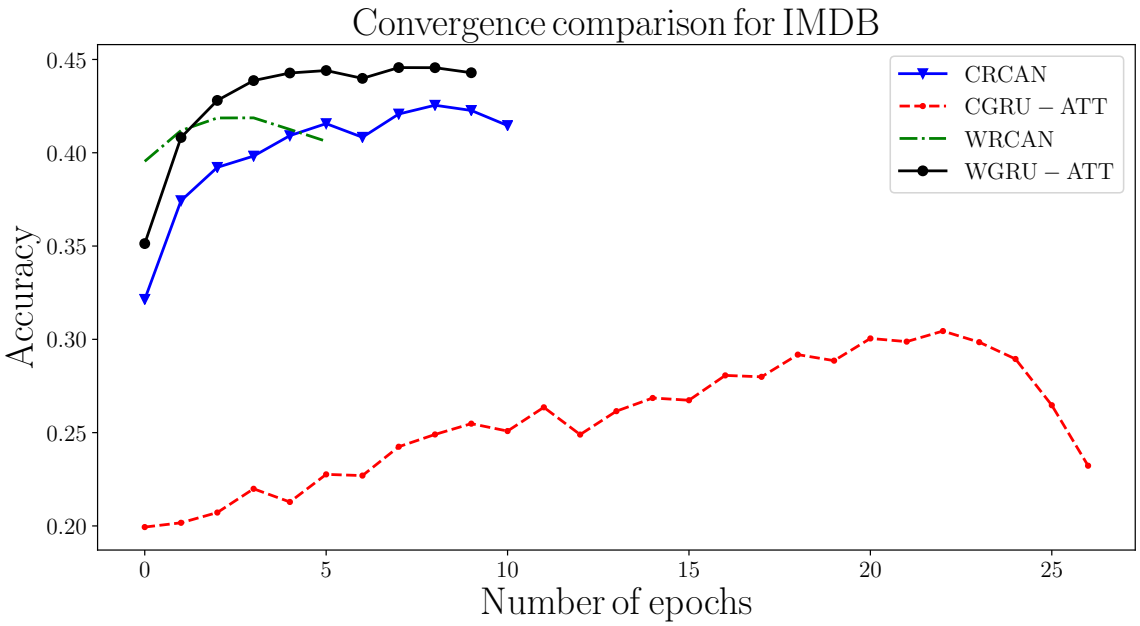


Figure 4.3: Comparison of convergence speed for the character level models (CRCAN and CGRU-ATT) and word level models (WRCAN and WGRU-ATT) for **IMDB** dataset.

4.2.2 Learning performance of Kim et al. convolution neural vs Yang et al. recurrent neural approach

We also compare the learning performance amongst the models, the comparison we show is the ability of the parallel convolutional layer based custom feature construction method proposed by Kim et al. against stratified GRU layers proposed by Yang et al., to learn overexposed to the information signal from the instances. As

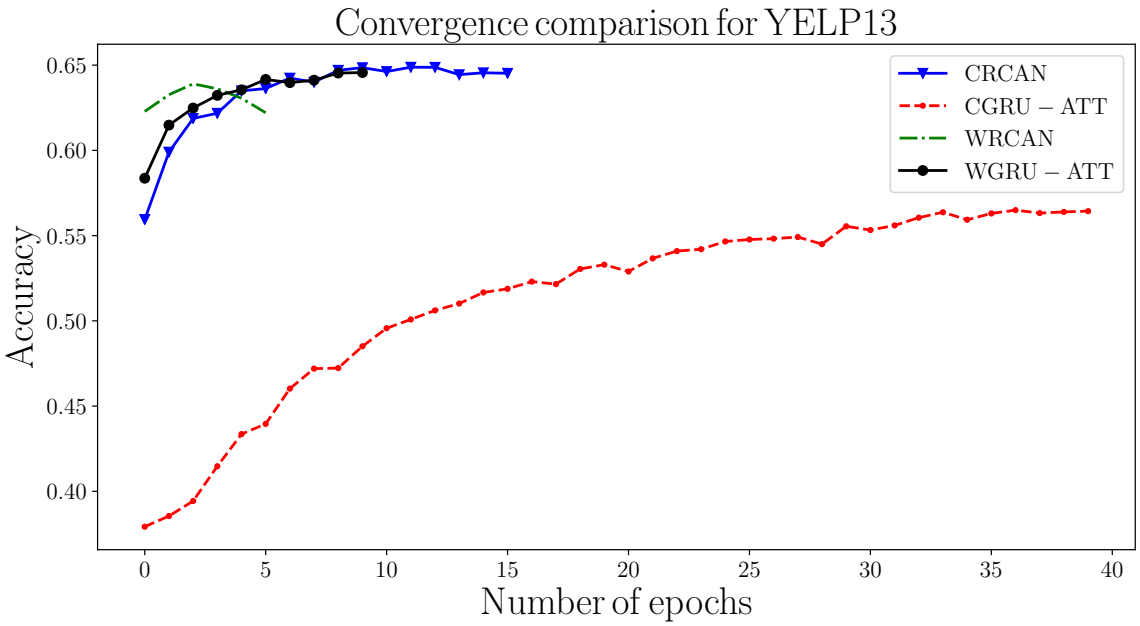


Figure 4.4: Comparison of convergence speed for the character level models (CRCAN and CGRU-ATT) and word level models (WRCAN and WGRU-ATT) for **YELP13** dataset.

BERT methods load the pre-trained parameters, thus we do not include the BERT variants in the comparison. Over first epoch, we setup the breakpoints where we compute the classification performance over validation set, which shows the information gained by the model and that we call it as the experiment. We use validation F1 score to gauge the performance and execute experimental runs for logging the values after every 100 instances of training. The breakpoint measures are then plotted against the number of training instances exposed to the models as shown in fig. 4.1. Later on training over more epochs, the neural model performance improves and we evaluate the performance over multiple timestamps. We see the point of convergence where performance improvement remains approximately the same and later

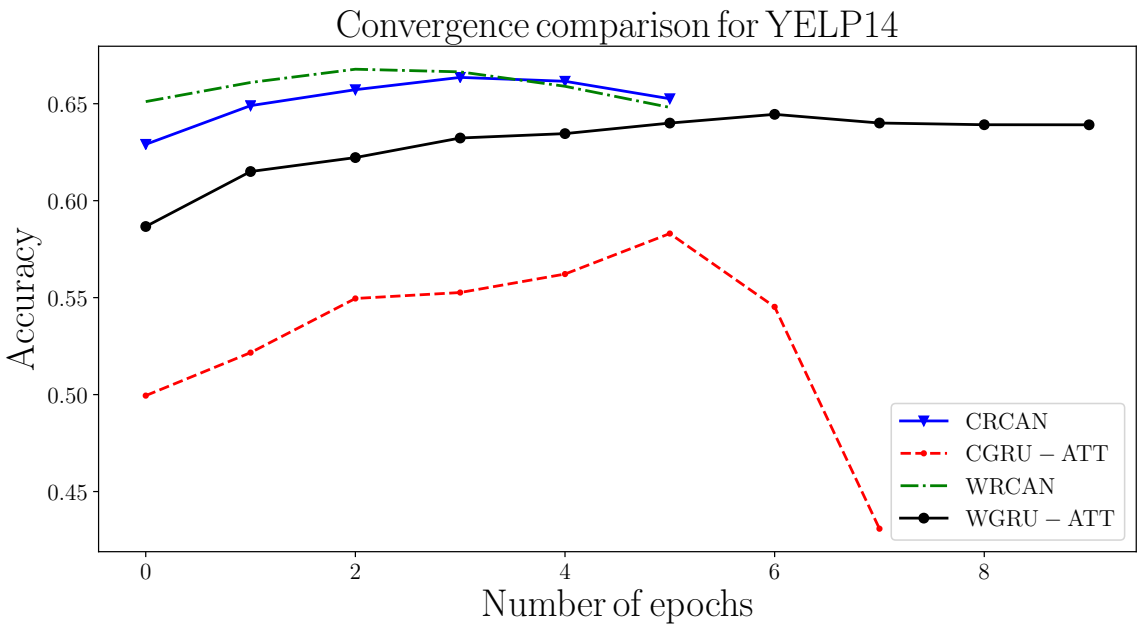


Figure 4.5: Comparison of convergence speed for the character level models (CRCAN and CGRU-ATT) and word level models (WRCAN and WGRU-ATT) for **YELP14** dataset.

it declines.

Figure 4.1 and fig. 4.2, show the performance of learning velocity in terms of F1 score with the percentage of instances for single epoch training over the IMDB and YELP-13 datasets. Based on these two figures, we see that among the character-level models i.e. CRCAN is trained faster than CGRU. Similar behaviour is observed for the word-level models, WRCAN compared against WGRU-ATT. Thus, the information learned and the convergence of the convolution models is faster than that of GRU based models. Moreover, CRCAN model shows competitive learning performance with the baseline HAN model as referred in fig. 4.1 and fig. 4.2, the margin of those models is relatively closer than others. The word-level HAN model WAGRU

Datasets	**WGRU-ATT**	**WRCAN**	**CRCAN**	**CGRU-ATT**
IMDB	8,162,610	7,423,817	3,489,517	229,810
Yelp 13 & 14[2]	8,162,105	7,423,312	3,489,012	229,305

Table 4.7: The total number of learning parameter for dataset IMDB and YELP for four models WGRU-ATT, CGRU-ATT, WRCAN and CRCAN

Dataset	**BERT-AN(a)**		**BERT-AN(b)**	
	Trainable	**Total**	**Trainable**	**Total**
IMDB	22,119,690	110,370,372	7,935,374	110,361,800
YELP 13 &14	22,118,405	110,369,087	7,934,869	110,361,295

Table 4.8: The total number of the trainable and the total parameters required for the BERT-AN(a) and BERT-AN(b)

has the number of training parameters than character-level CRCAN, furthermore, the word-level model has a large set of embedding lookups whereas in the character-level model we have extra stratified layer but the lower dimension of the embedding lookups. Even with lower learning parameters as referred in table 4.7, we see the relatively faster learning rate in CRCAN model.

We see the similar trends in terms of convergence speed for the models which stems from our approach's faster learning ability. We determined a model to have converged in training if its per-epoch validation accuracy decreased for two epochs in a row. Using this criterion in fig. 4.3, fig. 4.4 and fig. 4.5 as stopping condition, we

note the convergence point as higher accuracy acquired at a certain epoch noted. We see the evidence of our approach achieving convergence point with applied criteria faster than baseline models in both word-level and character-level methods. We do a comparison of CGRU-ATT with CRCAN method for character level methods and WGRU-ATT method with WRCAN method. Notice also that not only does our approaches converge faster, but it converges to a higher validation accuracy. As the BERT is recent approach and suppose to require lesser training but with extension to the character, level makes the BERT method complex to train and increase the number of learning parameters as can be seen in table 4.8 and table 4.7, whereas in CRCAN the convolution kernel is smaller and relatively faster to train.

The convergence speed of the word level HAN model has an analogous pattern with the CRCAN model for all the dataset, which is again a piece of supplementary evidence which motivates the use of parallel CNNs. We also observe that the performance of CGRU for the learning velocity and convergence speed do not compete with any of the other models. Thus, in results shows that tri-stratified layer of GRUs (CGRU-ATT) takes longer period in the exposure of the information to learn from coarser level representation, than that of WRCAN and WGRU-ATT which has analogous performance with CRCAN.

4.2.3 More extracted information, relatively fewer lexical representation

Noting that the word level stratified HAN model requires heavy vocabulary size and lexical learned over the training which makes it hard to connect the relational information using GRU. However, CRCAN model requires significantly lower lexical representation learned, also give control to cast features specific to the characteristic of n-grams windows. Finally, the feature construction method provides higher information flow which is parameterized with the number of kernels.

We perform an experiment where we compute the reconstruction of the words possible over limiting certain characters in the vocabulary across the datasets. While training any word-level stratified models, we have a substantial vocabulary size. We limit the size of this vocabulary with only including a set of stop words, we call it a modified vocabulary. We apply the same mechanism with characters vocabulary, as an instance we have our modified vocabulary $C = \{\text{'e'}, \text{'i'}, \text{'r'}, \text{'t'}, \text{'v'}, \text{'}\langle\text{UNK}\rangle_c\text{'}\}$. We look for all the character for the unique word "retriever" present in the modified character vocabulary C. We count all such possible occurrence for the characters of the unique words, we call it an effective vocabulary. The counts we observe for the effective vocabulary are shown in table 4.8. We see that effective vocabulary is 99.9% of the word vocabulary. We look for all the character for the unique word "retriever" present in the modified character vocabulary C. We count all such

[2]As the dataset YELP 13 & 14 have the same document characteristics, the total number of learning parameters in the model remains the same.

Datasets	**Effective Vocab**	**Word Vocab**
IMDB	115186	115193
Yelp 13	184850	184867
Yelp 14	414406	414748

Table 4.9: The size counts of effective vocabulary and the word vocabulary for the Datasets

possible occurrence for the characters of the unique words, we call it an effective vocabulary. The counts we observe for the effective vocabulary are shown in table 4.9. We see that effective vocabulary is 99.9% of the word vocabulary. To support the above claim, we show visual evidence which can be referenced from fig. 4.6. In the figure, we show the energy space acquired by the word feature representation from the WGRU-ATT model and CRCAN model. Recapturing that CRCAN models synthesize the word features from the character features, given that the vocabulary size of characters is significantly lesser than the vocabulary of words. The points in the graph show the reduced dimension of the feature space using T-SNE algorithm.

We experiment with projecting the word representation of both methods to the same space using TSNE algorithm. We utilize a document from the validation set and isolate 150 unique words, extract word representations for those words and projected them into the same space with TSNE algorithm setting number of components as 2 and initializing the parameters of it with zero random states.

We observe the projected symbols for the words and as we notice that word

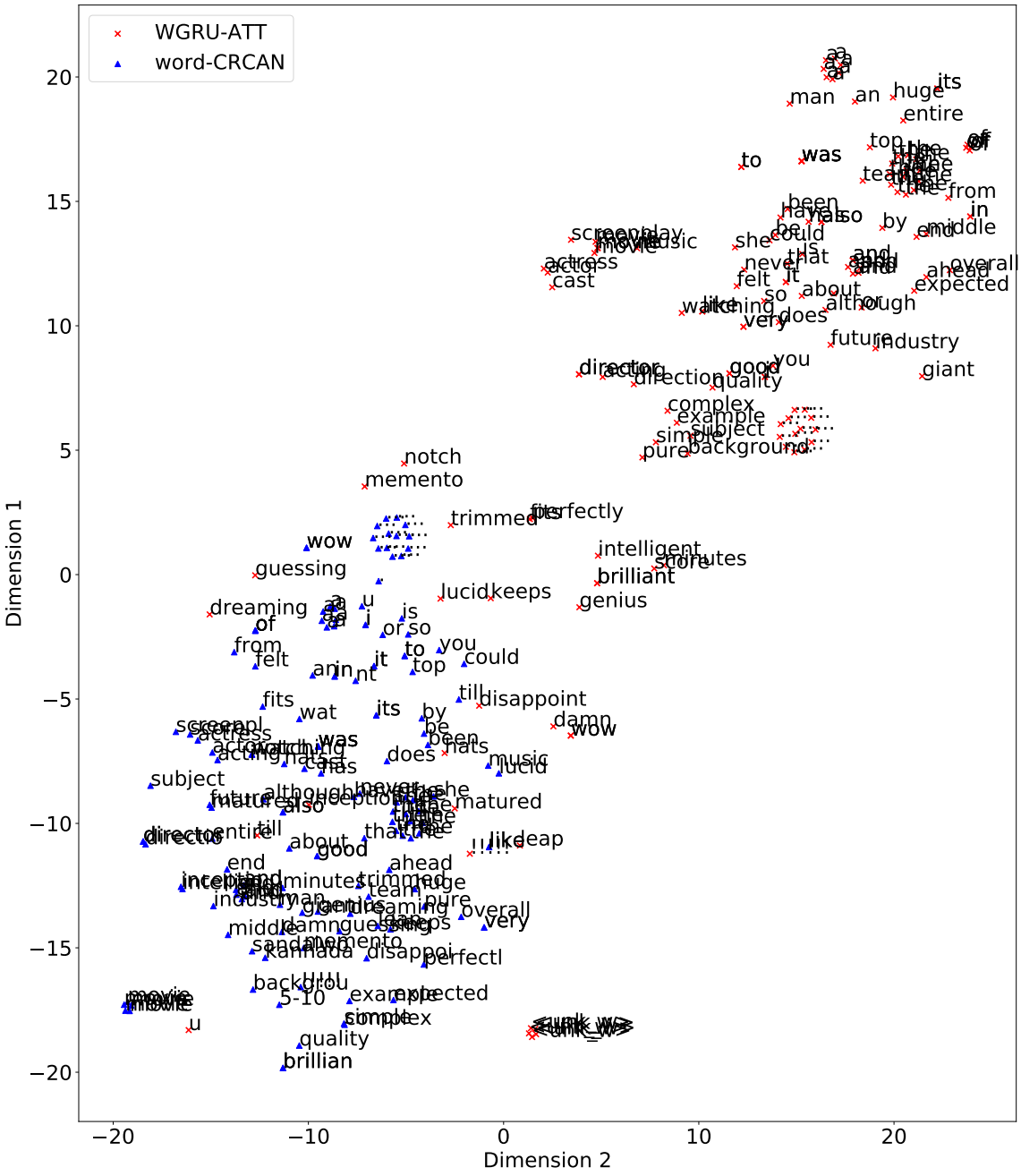


Figure 4.6: Projection of the word representation from the WGRU-ATT model and constructed word representation from CRCAN model into same reduced space

representations are widely scattered than that of the constructed word representation. The word representation is learned as embedding vectors in the WGRU-ATT model which are time series invariant. In CRCAN method, the word features are constructed using sequential information of the characters. Thus, we observe that word representation from CRCAN method is scattered denser than that of WGRU-ATT method. We also see that words in word representations of CRCAN method have similar words gathering closer, which is not the case in the word representation of WGRU-ATT method. Thus along with evidence of rich context-aware word information content, we also observe the extra words which are in the place of an unknown tag are available with CRCAN method. Thus not only contextual features but the high volume of the features representation of the words in the effective vocabulary is present, which results in boosting of the GRU with attention method's performance.

4.3 Summarised benefits and observations of our approach

Our stratified neural approach for document classification provides several benefits, we summarize them in this section. We show that employing the Kim et al. [16] convolutions results in faster learning rate and convergence. Also, the convolution kernels are implemented to execute in parallel in the processor which makes computation faster. Information-rich feature construction is resulted using convolution from coarser level lexical representation. We show visual evidence which

[2]As the dataset YELP 13 & 14 have the same document characteristics, the total number of learning parameters in the model remains the same.

depicts the quality of the word features constructed. Thus with the character vocab 99.9% of effective vocabulary reconstruction of the complete word vocabulary. Employing the convolution method, we enable the flexibility in defining the dimension of the features and can control the profile of the feature casting using a window of the kernel.

The performance measure comparison in table 4.3, 4.4 and 4.5 have average improvement of +8.8% on average which is relative to CGRU-ATT baseline model over the accuracy, precision, recall and F1 score. Our approach provides the flexibility of casting the neural models analogous to the stratification of the document. Example, the document contains paragraphs, sentences, words, characters. Similarly, we can extract all the hidden semantic information within different stratification of the document. Finally, we show the comparative studies shows the CRCAN method outperforms BERT-AN(b) method with +2.25% and BERT-AN(a) method with +1.6%.

Chapter 5: Conclusion and Future Work

5.1 Conclusion of the approach and results inline to the hypothesis

We propose hybrid neural architecture analogous to input the stratified structure of the document, consist of parallel convolution kernel-based feature construction in coherence with stacked GRU attention based layers. We discussed the performance analysis over different metrics such as accuracy, precision, recall and F1 score using IMDB and YELP datasets, which is strong evidence for hypothesis 1 and 2. The quantitative studies show the promising results of our approach which also outperforms the recent state of the art method, such as BERT extensions [8]. The quantitative performance of our method and the architecture of each module gives the confidence to claim for the requirement of the stratified learning for the task of document classification. Thus, we call our approach a modular approach which can be modified as analogous to the document stratification.

Moreover, our observation also includes the experiments reckon hypothesis 3 and we show the learning rate and convergence rate faster with our methods in both word-level and character-level methods. Further, we show the qualitative analysis of the word features constructed using convolution method in CRCAN method, the rich feature characteristics support the hypothesis 4. We show that 99% of word

vocabulary features are constructed which we call it as effective word vocabulary from the character level features.

Finally, we summarize the inherited layer benefits associated with our approach which enables fast convergence, the high volume of information extraction, elevated performance, flexible feature construction and wide set of generalized embedding for the core tasks.

5.2 Future work and contributions plans

One of the major extension of this work is to train the model in unsupervised learning methods. We can break down one large document into two sub-document and label as positive if the two subdocuments entail else we label as zero. We can train the model for the entailment classification and generated the pre-trained parameters similar to the BERT method, which can further be used for the downstream tasks.

The task of classification for the stratified CRCAN method can be used and encoder for the task of machine translation of the documents, where we can apply methods for feature uncoiling methods in decoder model in the stratified method.

We can also utilize the CRCAN approach in the recommendation system, where the recommended units are dependent on the more than one entities example product reviews entailed with the product details and images, where idea stems from the grounding multi-modal signals into a common representation and extract the contextual information from the multi-modal information, utilize the extracted

features for the ranking task.

Appendix A: Extended Comparison for CRCAN and WRCAN

Following are the tables for the extended comparison of our models with baseline. We broadly compare the character-level models with word-level models (table A.2, table A.4 & table A.6) . Moreover, we show tables for the BERT method models with our character level approaches (table A.1, table A.3 & table A.5).

Metrics	**CGRU-ATT**	**CRCAN**	**BERT-AN(a)**	**BERT-AN(b)**
F1 Score	22.77	**46.61**	43.70	37.31
Precision	21.62	**36.83**	33.19	31.73
Recall	24.95	**38.08**	34.80	33.04
Accuracy	31.46	**41.56**	40.51	38.45

Table A.1: The classification performance comparison of CRCAN, extended BERT models - BERT-AN(a) and BERT-AN(b) and CGRU-ATT - character-level HAN models over **IMDB** dataset (test set)

Metrics	**WGRU-ATT**	**WRCAN**	**CGRU-ATT**	**CRCAN**
F1 Score	**47.37**	42.93	22.77	46.61
Precision	**39.32**	36.29	21.62	36.83
Recall	**39.94**	37.25	24.95	38.08
Accuracy	**44.53**	41.87	31.46	41.56

Table A.2: The classification performance comparison of CRCAN, WRCAN, WGRU-ATT - word level HAN model (Baseline) and CGRU-ATT - character-level HAN models over **IMDB** dataset (test set)

Metrics	**CGRU-ATT**	**CRCAN**	**BERT-AN(a)**	**BERT-AN(b)**
F1 Score	26.65	**31.73**	30.40	30.68
Precision	26.62	**31.10**	30.09	29.80
Recall	26.71	**30.75**	29.99	29.52
Accuracy	57.74	**64.85**	63.12	63.51

Table A.3: The classification performance comparison of CRCAN, extended BERT models - BERT-AN(a) and BERT-AN(b) and CGRU-ATT - character-level HAN models over **YELP-13** dataset (test set)

Metrics	**WGRU-ATT**	**WRCAN**	**CGRU-ATT**	**CRCAN**
F1 Score	30.37	31.16	26.65	**31.73**
Precision	30.58	30.58	26.62	**31.10**
Recall	**31.14**	30.39	26.71	30.75
Accuracy	64.48	63.77	57.74	**64.85**

Table A.4: The classification performance comparison of CRCAN, WRCAN, WGRU-ATT - word level HAN model (Baseline) and CGRU-ATT - character-level HAN models over **YELP-13** dataset (test set)

Metrics	**CGRU-ATT**	**CRCAN**	**BERT-AN(a)**	**BERT-AN(b)**
F1 Score	27.99	**32.35**	31.52	32.06
Precision	26.73	**31.74**	30.83	31.59
Recall	26.15	**31.44**	30.37	31.38
Accuracy	58.30	66.35	64.95	**66.50**

Table A.5: The classification performance comparison of CRCAN, extended BERT models - BERT-AN(a) and BERT-AN(b) and CGRU-ATT - character-level HAN models over **YELP-14** dataset (test set)

Metrics	**WGRU-ATT**	**WRCAN**	**CGRU-ATT**	**CRCAN**
F1 Score	31.22	32.29	27.99	**32.35**
Precision	31.11	30.58	26.73	**31.74**
Recall	31.03	**31.58**	26.15	31.44
Accuracy	64.34	**66.77**	58.30	66.35

Table A.6: The classification performance comparison of CRCAN, WRCAN, WGRU-ATT - word level HAN model (Baseline) and CGRU-ATT - character-level HAN models over **YELP-14** dataset (test set)

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Fidel Cacheda, Vassilis Plachouras, and Iadh Ounis. A case study of distributed information retrieval architectures to index one terabyte of text. *Information processing & management*, 41(5):1141–1161, 2005.

[3] Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, pages 9539–9549, 2018.

[4] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, volume 161175. Citeseer, 1994.

[5] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[6] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*, 2016.

[7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Javid Ebrahimi and Dejing Dou. Chain based rnn for relation classification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1244–1249, 2015.

[10] Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. Contextual lstm (clstm) models for large scale nlp tasks. *arXiv preprint arXiv:1602.06291*, 2016.

[11] Ruining He, Chunbin Lin, Jianguo Wang, and Julian McAuley. Sherlock: sparse hierarchical embeddings for visually-aware one-class collaborative filtering. *arXiv preprint arXiv:1604.05813*, 2016.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Aaron Jaech, George Mulcaire, Shobhit Hathi, Mari Ostendorf, and Noah A Smith. Hierarchical character-word models for language identification. *arXiv preprint arXiv:1608.03030*, 2016.

[14] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[15] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[16] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016.

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 (11):2278–2324, 1998.

[18] David D Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Third annual symposium on document analysis and information retrieval*, volume 33, pages 81–93, 1994.

[19] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *arXiv preprint arXiv:1812.09449*, 2018.

[20] Leo Liberti, Carlile Lavor, Nelson Maculan, and Antonio Mucherino. Euclidean distance geometry and applications. *Siam Review*, 56(1):3–69, 2014.

[21] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[23] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[24] Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *SIGIR*, volume 97, pages 67–73, 1997.

[25] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[26] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1063–1072. International World Wide Web Conferences Steering Committee, 2018.

[27] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[28] Hetal C. Chaudhar K. P.Wagh P.N.Chatur. Search results clustering using tf-idf based apriori approach: A survey paper. *International Journal of Engineering and Computer Science*, 4(01), Jan. 2015. URL `http://www.ijecs.in/index.php/ijecs/article/view/809`.

[29] Hinrich Schütze, David A Hull, and Jan O Pedersen. A comparison of classifiers and document representations for the routing problem. In *Annual ACM conference on Research and Development in Information Retrieval-ACM SIGIR*. Citeseer, 1995.

[30] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Hierarchical neural network generative models for movie dialogues. *arXiv preprint arXiv:1507.04808*, 7(8), 2015.

[31] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[32] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[33] K Soumya George and Shibily Joseph. Text classification by augmenting bag of words (bow) representation with co-occurrence feature. *IOSR J. Comput. Eng*, 16(1):34–38, 2014.

[34] Shane Storks, Qiaozi Gao, and Joyce Y Chai. Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. *arXiv preprint arXiv:1904.01172*, 2019.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

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

[37] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, pages 2915–2921, 2017.

[38] Erik Wiener, Jan O Pedersen, Andreas S Weigend, et al. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th annual symposium on document analysis and information retrieval*, volume 317, page 332. Las Vegas, NV, 1995.

[39] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[40] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

[41] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1-2):69–90, 1999.

[42] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, page 35, 1997.

[43] Yiming Yang and John Wilbur. Using corpus statistics to remove redundant words in text categorization. *Journal of the American Society for Information Science*, 47(5):357–369, 1996.

[44] Yiming Yang, Xin Liu, et al. A re-examination of text categorization methods. In *Sigir*, volume 99, page 99, 1999.

[45] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, pages 1480–1489, 2016.

[46] Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*, 2017.

[47] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[48] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[49] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

[50] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.

[51] Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612, 2015.

[52] Si Zuo and Zhimin Xu. A hierarchical neural network for sequence-to-sequences learning. *arXiv preprint arXiv:1811.09575*, 2018.