Deep Learning Methods for Short, Informal, and Multilingual Text Analytics

Muhammad Haroon Shakeel

Supervisor: Dr. Asim Karim

Department of Computer Science
Lahore University of Management Sciences (LUMS), Pakistan

This dissertation is submitted for the degree of
Doctor of Philosophy

June 2020

Certificate of Approval

This is to certify that the research work presented in this thesis, entitled **"Deep Learning Methods for Short, Informal, and Multilingual Text Analytics"** was conducted by **Mr. Muhammad Haroon Shakeel** under the supervision of **Dr. Asim Karim**. No part of this thesis has been submitted anywhere else for any other degree. This thesis is submitted to the **Department of Computer Science, Lahore University of Management Sciences (LUMS)** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in the field of **Computer Science**.

Student Name: Muhammad Haroon Shakeel Signature:_____

Final Defense Committee:

(a) **Internal Examiner 1:** Dr. Mian M. Awais Signature:_____
Department of Computer Science
Lahore University of Management Sciences

(a) **Internal Examiner 2:** Dr. Imdadullah Khan Signature:_____
Department of Computer Science
Lahore University of Management Sciences

(a) **Internal Examiner 3:** Dr. Murtaza Taj Signature:_____
Department of Computer Science
Lahore University of Management Sciences

(a) **External Examiner 1:** Dr. Zubair Khalid Signature:_____
Department of Electrical Engineering
Lahore University of Management Sciences

(a) **External Examiner 2:** Dr. Faisal Kamiran Signature:_____
Department of Computer Science
Information Technology University

Supervisor Name: Asim Karim Signature:_____

Department Chair: Ihsan Ayyub Qazi Signature:_____

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgments.

Muhammad Haroon Shakeel
June 2020

Plagiarism Undertaking

I solemnly declare that research work presented in the thesis titled **"Deep Learning Methods for Short, Informal, and Multilingual Text Analytics"** is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Lahore University of Management Sciences (LUMS) towards plagiarism. Therefore, I as an author of the above titled thesis, declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if am found guilty of any formal plagiarism in the above titled thesis even after award of PhD degree, the University reserves the rights to withdraw/revoke my PhD degree and that HEC and the University has the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized thesis.

Signature: _____

Muhammad Haroon Shakeel
June 2020

I would like to dedicate this thesis to my loving parents and my uncle Barkat Ali Chaudhry. It is due to their support and encouragement that I have achieved this much and standing where I am today. Secondly, I would like to dedicate my work to my supervisor, Dr. Asim Karim, for helping, assisting, and guiding me through each and every step during my challenging stretch.

Acknowledgements

I would like to acknowledge the support of my supervisor Dr. Asim Karim, my co-supervisors Dr. Imdadullah Khan and Dr. Murtaza Taj who aided me to achieve what I wanted to do, and lent a hand to me to reach such significant outcomes. I am also gratified to my colleagues Numan Khurshid, Waseem Abbas, Mazher Iqbal, Mohbat Tharani, and Syeda Farheen Naz who helped with my research. Finally, I would like to thank Higher Education Commission of Pakistan (HEC) for funding my research under HEC Indigenous Scholarship Program.

Author Biography

Muhammad Haroon Shakeel is in the fifth year of his Ph.D. in the Department of Computer Science at Lahore University of Management Sciences (LUMS), Pakistan. In his Ph.D. dissertation, he focuses on performing text analytics on short, informal, and multilingual text using deep learning methods. He is also interested in natural language processing, computer vision, big data analytics, machine learning, and deep learning.

He received his bachelor's degree with double majors in Information Technology and Business Administration from International Islamic University, Islamabad, Pakistan, and a master's degree in Computer Science from COMSATS Institute of Information Technology, Islamabad, Pakistan.

He worked in collaboration with Victoria University, Melbourne, Australia, on the detection of domestic violence from online text. He is also involved in multiple projects at Computer Vision and Graphics Lab (CVGL) and Data Analytics Lab at LUMS, mainly focusing on big data analytics and medical imaging. His current placement is in the Knowledge and Data Engineering (KADE) lab at LUMS.

He loves to travel, is a hardcore computer gamer, and does photography as a hobby.

Publications

- **Publications From Thesis**

1. **Muhammad Haroon Shakeel**, Asim Karim, Imdadullah Khan, *"A Multi-cascaded Model with Data Augmentation for Enhanced Paraphrase Detection in Short Texts"*, Information Processing and Management (IPM), Volume 57, Issue 3, 2020.
2. **Muhammad Haroon Shakeel**, Asim Karim, *"Adapting Deep Learning for Sentiment Classification of Code-Switched Informal Short Text"*, 35th ACM/SIGAPP Symposium on Applied Computing (ACM-SAC), April 2020.
3. **Muhammad Haroon Shakeel**, Safi Faizullah, Turki Alghamidi, Imdadullah Khan, *"Language Independent Sentiment Analysis"*, IEEE International Conference on Advances in the Emerging Computing Technologies (AECT), February 2020.
4. **Muhammad Haroon Shakeel**, Asim Karim, Imdadullah Khan, *"A multi-cascaded deep model for bilingual SMS classification"*, 26th International Conference on Neural Information Processing (ICONIP), December 2019.

- **Other Publications**

1. Sarwan Ali, **Muhammad Haroon Shakeel**, Imdadullah Khan, Safiullah Faizullah, Muhammad Asad Khan, *"Predicting Attributes of Nodes Using Network Structure"*, ACM Transaction on Intelligent Systems and Technology (ACM-TIST), June 2020.
2. Waseem Abbas, **Muhammad Haroon Shakeel**, Numan Khurshid, Murtaza Taj, *"Patch-based Generative Adversarial Network Towards Retinal Vessel Segmentation"*, 26th International Conference on Neural Information Processing (ICONIP), December 2019.
3. Sudha Subramani, Sandra Michalska, Hua Wang, Jiahua Du, Yanchun Zhang, **Haroon Shakeel**, *"Deep Learning for Multi-class Identification from Domestic Violence Online Posts"*, IEEE Access 7, pp. 46210-46224, January 2019.
4. Waseem Abbas, Numan Khurshid, **Muhammad Haroon Shakeel**, *"Medical Image Compression Using DDCT, Ripplet Transform and SPIHT"*, International Journal of Scientific and Engineering Research (IJSER) 9 (6), June 2018.

- **Under Review**

1. Muhammad Ahmad, **Muhammad Haroon Shakeel**, Sarwan Ali, Imdadullah Khan, Arif Zaman, Asim Karim, *"Efficient Data Analytics on Augmented Similarity Triplets"*, The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD) - Journal Track.

Abstract

The popularity of social media platforms and knowledge sharing websites has tremendously increased the amount of user-generated textual content. Such content is usually short in length and is often written informally (e.g., improper grammar, self-created abbreviations, and varying spellings). It is also influenced by local languages and mix multiple languages mid-utterance, a phenomenon known as code-switching. Traditional text analytics and natural language processing (NLP) approaches perform poorly on short, informal, and multilingual text as compared to well-written longer documents because of the limited context and language resources available for learning. In recent years, deep learning has produced enhanced results for many NLP tasks. However, these approaches have some major shortcomings: (1) they are tailored for specific problem settings (e.g., short text or informal languages) and do not generalize well to other settings, (2) they do not exploit multiple perspectives and resources for effective learning, and (3) they are hampered by smaller training datasets.

In this research, we present methods and models for effective classification of user-generated text with a specific application to English and Roman Urdu short and informal text. We present a novel multi-cascaded deep learning model (*McM*) for robust classification of noisy and clean short text. McM incorporates three independent CNN and LSTM (with and without soft attention) cascades for feature learning. Each cascade is responsible for capturing a specific aspect of natural language. The CNN based cascade extracts *n*-gram information. The LSTM based cascade with soft-attention “highlights" the task-specific vital words. The third LSTM based cascade captures long-term dependencies of the text. Each cascade is locally supervised and is trained independently. The deep representations learned by each cascade are forwarded to a discriminator for final prediction. As a whole, the architecture is both deep and wide, and is versatile to incorporate learned and linguistic features for robust text classification.

We evaluate the effectiveness and generality of our model on three different text analytics problems. First, we show the efficacy of our model for the problem of paraphrase detection. This is a binary classification problem in which pairs of texts are labeled as either positive (paraphrase) or negative (non-paraphrase). While deep models produce a richer text repre-

sentation, they require large amounts of data for training purposes. Getting additional pairs of texts annotated in a crowd-sourced setting is costly. Thus, for this particular task, we also develop a novel data augmentation strategy, which generates additional paraphrase and non-paraphrase annotations reliably from existing annotations. The augmentation procedure involves several steps and a parameter through which the degree of augmentation can be tuned. We evaluate our model and data augmentation strategy on three benchmark datasets representing both noisy and clean texts. Our model produces a higher predictive performance on all three datasets beating all previously published results on them.

Second, we show the usefulness of McM for the task of multi-class classification of bilingual SMS. Our goal is to achieve this without any prior knowledge of the language, code-switching indication, language translation, normalizing lexical variations, or language transliteration. For this purpose, we develop and make publicly available a 12 class large-scale dataset. The texts in this dataset contain English as well as Roman Urdu, a distinct informal regional dialect of communication that uses English alphabets to write Urdu. Our model achieves greater robustness as compared to the baseline model on this dataset.

Third, we demonstrate the utility of the proposed model for the task of sentiment classification in code-switched tweets. For this purpose, we develop another short text dataset, namely MultiSenti, that is code-switched between Roman Urdu and English languages. The proposed model McM outperforms four baseline models on the MultiSenti dataset in terms of predictive accuracy. We also study the feasibility of adapting language resources from English and learning domain-specific word embeddings in Roman Urdu for multilingual sentiment classification.

This research highlights the power of multi-perspective feature learning and data augmentation for short and informal text classification and takes us a step closer to language-independent text analytics.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

With the proliferation of microblogs, knowledge sharing forums, and social networks, the user-generated textual content is being produced in abundance. This content can be about virtually anything including sports, politics, hardware, software, products, events, etc. With the increasing amount of user-generated content came the need by the companies, politicians, advertisers, service providers, governments, analysts, and researchers to automatically mine, analyze, and classify this textual content for different uses.

Due to limitation of the number of characters or words, the text produced under question-answer forums, microblogs, social media platforms, and short message service (SMS) is short in length [9, 25, 8]. Furthermore, social media posts such as Tweets or SMS, are commonly written in freestyle and informal verbatim (e.g., language irregularities, improper grammar, self-created acronyms, varying spellings, jargon). Such contents are referred to as “noisy texts". Moreover, these texts are also influenced by local languages, where the writer utilizes two or more languages simultaneously, a factor known as multilingualism [22]. Consequently, alternation of multiple languages in a single text, a phenomenon known as code-switching, gives birth to new informal dialects of communication [96]. For instance, Urdu is the National language of Pakistan, while English is treated as the official language of the country. This leads to the development of a distinct dialect of communication known as “Roman Urdu", which utilizes English alphabets to write Urdu.

Traditionally, in Natural Language Processing (NLP), texts are considered as a bag of words where only the appearance of a word in the text is accounted for and its position or the order in the text does not matter. Therefore, classifiers implemented on this representation of text mainly try to find keywords that could be used to make correct predictions. This approach works well with longer texts that follow standard spellings and grammar (also known as clean texts) but performs poorly for short and noisy ones [71]. Since short and informal texts hold limited context with unique characteristics, it make them difficult to

Table 1.1 Bag of words representation of short and informal text

who	is	Trump	?	will	Hilary	lose	hilry	looz
1	1	1	1	0	0	0	0	0
0	0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	1	1

handle [71]. For instance, the text representation using the traditional bag of words model becomes high dimensional and very sparse as there are very few words that are common amongst text in a dataset. Let's consider the following three texts. (1) *who is Trump ?* (2) *will Hilary lose ?* (3) *will hilry looz ?* These texts will produce a bag of word representation as presented in Table 1.1. In this example, each row represents one particular text, while each column shows the word in the text. Note that as the number of texts in the table grows, the first row would become sparser as there would be very few words shared with the first text. Similarly, informal and non-standard spellings of third text produce unnecessary two dimensions for all texts (shown in the last two columns), consequently increasing the sparsity. Thus, traditional text analytics and NLP approaches developed for formal, clean, and longer texts are not well-suited for user-generated short, informal, and code-switched ones and tend to perform poorly [16, 95].

In recent years, deep learning has been extensively used to understand and model the short text for many NLP and text analytics tasks [1, 4, 3]. One of the benefits of deep learning approaches is that it learns richer and denser representations of text as compared to bag of words models and does not require sophisticated feature engineering to model the richness of human language [14]. Existing approaches, however, have some major shortcomings. (1) They are tailored to specific problem settings. The strategies that work well for formal short text do not give a good performance against informal, noisy, and code-switched text and vice versa. (2) They do not exploit multiple linguistic perspectives and resources for robust learning. (3) They are data-hungry and hampered by smaller training datasets [16].

In this thesis, we present methods and models for effective classification of user-generated text with a specific application to English and Roman Urdu short and informal text. To this end, first, we present a novel multi-cascaded deep learning architecture called *McM* for robust classification of both noisy and clean short text across multiple text analytics tasks. McM exploits multiple aspects of linguistics through three independent Convolutional Neural Network (CNN) and Long Short-term Memory (LSTM) based cascades for feature learning. The first, CNN based cascade, extracts the local features in the form of important *n*-grams. The second LSTM based cascade "highlights" the task-specific most important words in the text. While the third LSTM based cascade learns long-term text dependencies. Each of these

cascades is locally supervised with task-specific labels and is trained independently. The text representations learned by each cascade is then forwarded to a discriminator network for final prediction. As a whole, the proposed model is both deep and wide while being flexible to incorporate both learned and hand-crafted linguistic features to form a robust text classification system.

We evaluate the effectiveness and generalization power of the proposed model on three text analytics tasks. First, we show the efficacy of our model for paraphrase identification. This is treated as a binary classification problem where a pair of texts is marked as paraphrase if both texts are semantically identical or non-paraphrase otherwise. While deep models produce a richer text representation, they are data-hungry and require large amounts of labeled data for learning [16]. On the other hand, obtaining more labeled data in a human-based computation setting (e.g., crowd-sourcing) incur monetary cost [97]. Thus, for this particular task, we also develop a novel systematic data augmentation strategy to generate additional paraphrase and non-paraphrase pairs from existing annotations reliably. The augmentation procedure is parameterized to control the degree of augmentation. We evaluate the proposed model and data augmentation method on three benchmark datasets, representing both noisy and clean texts. In our presentation, we show that McM produces a higher predictive performance, beating previously published results on all three datasets.

Secondly, we demonstrate the efficacy of McM on multi-class text classification of bilingual SMS. This shows that the proposed model is a general-purpose model and can be utilized across multiple text analytics tasks. We tend to achieve this without using any external knowledge base, language translation, code-switching indication, language transliteration, or lexical normalization. To this end, we develop a 12 class large-scale dataset in English as well as Roman Urdu. Our model achieves greater robustness as compared to the baseline model on this dataset.

Thirdly, we demonstrate the utility of the proposed model for the task of sentiment classification of code-switched tweets written in informal language. For this purpose, we develop another annotated dataset called “MultiSenti". The text in this dataset is code-switched between Roman Urdu and English languages. We compare the performance of McM with four baseline models and show that it outperforms them all in terms of predictive accuracy on MultiSenti. Additionally, we also study the feasibility of adapting the language resources from English and learning domain-specific word embeddings in Roman Urdu for this task.

The extensive experiments of this research lead us to conclude that McM can learn efficient representations from the limited context available for multiple text analytics tasks while being robust to noise, informal language, improper grammar, code-switching, or

dataset size. The findings highlight the power of multi-perspective feature learning and data augmentation and take us a step closer to language-independent text analytics.

1.1 Motivation

In recent years, there has been a tremendous increase in the user-generated short, informal, and multilingual textual content. Publicly available texts are produced in the form of posts on social media platforms, questions and answers on forums, comments, reviews of products, news headlines, and personal opinions, to name a few. The information content of this data is very rich and is of great utility to various sectors of society and the economy. Automatic text analytics on this content is of great value for companies, governments, manufacturers, retailers, media organizations, academia, and many other entities. For instance, with millions of users visiting question-answer websites, people inevitably ask questions with the same semantic meaning or intention, producing duplicates (paraphrases). On such sites, detecting a duplicate pair of short text is an essential task as it not only saves storage space on servers but also saves time for both the answer seekers and knowledge experts. Consequently, it also increases the efficiency of such forums and helps question answering systems to deliver focused answers to user's questions. Paraphrase identification also has applications in plagiarism detection, query ranking, information retrieval, and document summarization.

Similarly, in e-government practices, citizens provide feedback on different public services that they availed using social media or SMS. Manual identification of complaints from such texts is impractical. Thus, an automated system to identify the category of the text has an excellent utility for good governance. Such a system can be used by governments to reduce petty corruption and deficient delivery in services. Although English is the most common language used over the internet, there is a sizable body of texts that are written in various other informal languages. In Pakistan, for example, people tend to switch between English and Urdu languages mid-utterance. Existing linguistic resources and classification models are limited to the English language mainly. Thus, the research for Roman Urdu text classification lags behind due to the unavailability of large-scale annotated datasets and language-understanding models. Therefore, an annotated dataset and robust language model for multilingual text classification are of utmost importance.

Moreover, inferring opinions or feelings of the authors of the texts (referred to as sentiment classification) is also utilized by governments, manufacturers, advertisers, and organizations. Manufacturers and content producers can modify product designs or customize according to the market feedback inferred from sentiments of the reviews and comments provided by the users. This, in turn, can lead to higher customer satisfaction and hence

enhanced revenue. Advertisers and marketing agencies can use this information to adapt effective marketing strategies and optimize their budgets. Governments and policymakers could use it to predict social reactions to a policy proposal. Sports organizations and clubs can benefit from sentiment analysis on fan's comments to measure a player's contribution to a game and hence estimate player's worth.

Given the vast volume, velocity, and varying writing styles of such user-generated textual content, performing manual text analytics, even on a small subset of texts, is nearly impossible. Various deep learning methods using tools from machine learning, NLP, and computational linguistics have been developed to automate many text analytics tasks. However, these methods are tailored mainly for the English language and specific setting. Consequently, this limits the applicability of these methods to particular demographics and text analytics tasks. Hence, developing a deep learning model that is robust enough to perform multiple text analytics tasks in a multilingual text can open a broad avenue of applications.

1.2 Problem Statement and Research Objectives

We formalize the problem concerning three tasks described above as following.

We are given a document $X \in D$ (in case of multi-class or sentiment classification task) or a pair of texts $(W,X) \in D$ (in case of paraphrase identification task) and a set of labels $Y \in \{y_0, y_1, ..., y_n\}$. Labels here are task-specific. The document space D is comprised of short text, which can be in the English language, Roman Urdu, or a mixture of both English and Roman Urdu languages. Given a training set T of labeled texts $< X,Y >$ or $< W,X,Y >$, we wish to develop a supervised classification model or classification function M that can be used to map texts to the labels.

$$M : D \Rightarrow Y$$

Labels act as supervision for M. We denote the supervised learning method by γ and write $\gamma(T) = M$. The learning method γ takes training set T as input and returns the learned classification function M and is based on multiple deep learning cascades for learning multiple representations of a document. Once the learning is finished, we use M to predict label $\hat{Y}$ for unseen testing set τ of labeled texts $< U,Y >$ or a pair of texts $< U,V,Y >$). The quality of the model is evaluated through prediction accuracy of $\hat{Y} = Y \; \forall$ texts in τ.

The research objectives include the following research and experimental tasks.

- Conduct a Literature Review on paraphrase identification, multilingual text classification, and sentiment classification of informal code-switched short text.

- Development of large-scale annotated datasets for Roman Urdu short text classification and sentiment analysis.
- Development of robust paraphrase identification model for the English language that works well on both clean and noisy short text.
- Evaluating the robustness of the model for multi-class and sentiment classification of code-switched informal short text.
- Extensive error analysis and comparison with existing state-of-the-art models.

1.3 Contributions

The contributions of this thesis are summarized as follows. First, We introduce a novel deep learning model called McM based on Convolutional Neural Networks (CNN) and Long Short-term Memory (LSTM) to perform three tasks, i.e., (1) paraphrase identification, (2) multi-class classification of bilingual text, and (3) sentiment classification of code-switched text.

Second, for paraphrase identification task, we propose a novel data augmentation strategy to generate new and reliable annotated data from existing annotations to improve paraphrase identification performance. We also show the usefulness of this augmentation strategy to detect and correct potential errors in existing annotations. Moreover, we identify various linguistic features that are used to improve paraphrase identification accuracy further. We perform extensive experimentation and performance comparison on three benchmark paraphrase identification datasets. We show that our proposed approaches achieve current state-of-the-art on all three.

Third, we develop a large-scale dataset having texts in Roman Urdu and English languages to detect petty corruption and poor delivery of public services. The dataset is developed from SMS feedbacks of the citizens of Pakistan on different public services that they availed. The dataset has 12 classes, consisting of more than 0.3 million records, and has been made available publicly for future research. We modify McM to take one input (as opposed to two inputs in case of paraphrase detection) and show that the proposed model yields superior performance as compared to the baseline model.

Fourth, we develop a second dataset called MultiSenti from Tweets collected during the general elections of Pakistan in 2018 for sentiment classification of code-switched informal short text. It has been categorized into 3 classes i.e., *negative*, *positive*, and *neutral*, have more than $20,000$ records, and is made publicly available. We develop a novel model called

CNN-gram to solve the MultiSenti dataset and compare its performance with McM and four baseline models. We show that McM shows greater robustness across all experiments.

Fifth, we develop domain-specific embeddings in Roman Urdu, having 31,308 tokens and make them publicly available for future research. We demonstrate its usefulness in Roman Urdu text classification and sentiment analysis tasks.

Finally, we investigate the feasibility of adapting character-based pre-trained embeddings from the English language to Roman Urdu through extensive experiments.

Our experiments lead us to conclude that McM has greater robustness and generalization across multiple tasks, languages, and embeddings.

1.4 Thesis Organization

The rest of the thesis is arranged, such that chapter 2 presents background and a literature review of paraphrase identification, bilingual text classification, and sentiment classification of code-switched short texts. In chapter 3, details of the proposed paraphrase identification model, data augmentation strategy, and linguistic features are presented. In chapter 4, the work on multi-class classification of bilingual SMS is discussed. Chapter 5 introduces the MultiSenti dataset, feasibility of adapting resources from the English language for sentiment classification of code-switched informal short text, and usefulness of different embeddings. We give the concluding remarks and future work in chapter 6.

Chapter 2

Background and Literature Review

Over the last few decades, text analytics problems have been widely studied and addressed in many applications such as paraphrase identification, sentiment analysis, document categorization, etc. In this chapter, we present background, related work, and limitations of existing approaches with specific focus on text representation (feature engineering), paraphrase identification, and informal multilingual short text analytics.

2.1 Text Representation

Text classification is the task of assigning a predefined label to documents written in natural language. Preliminary works in this field focused on efficient preprocessing and feature representation. Most commonly, after preprocessing, the variable-length texts are first represented as fixed dimensional feature vectors. A classifier is then learned in this feature space (such as SVM with linear kernel[39, 52]) to find optimal class boundaries. The feature representation is sought to be richer in the sense that it preserves as much information as possible. Initial works in NLP used bag of words (BoW) vector models to represent text in a document, where only the presence or absence of a word was considered, and the order of words in the text was not accounted for. A key drawback of this approach is that it does not account for semantic and syntactic understanding [47]. For instance, there might be two documents with a high percentage of common words, but a different context and meanings. However, the BoW representation of both of the texts would be very similar. Furthermore, the words are treated as discrete atomic symbols, which do not provide useful information to the system regarding the relationship between the symbols that might exist. This means that the classification model can leverage very little of what it has learned about “cats" when it is processing data about “dogs" (such as that they are both animals, four-legged and

pets, etc.) [1]. Another limitation of representing words as unique discrete ids is that it leads to data sparsity, which consequently demands more data to train the classification models successfully. Therefore, it is necessary to take a step further from handcrafted features[51, 66] towards automatic learning of the text features [80].

In recent years, a major advance on document representation is the utilization of word embeddings based text representations [60, 67]. These representations are learned in supervised manner and produce a dense vectors, which can overcome some of the shortcomings of BoW models by capturing semantic information [59]. Embeddings models represent each word in a corpus as d dimensional continuous vector, where semantically similar words are mapped to nearby points in Euclidean space ('are embedded nearby each other') [6]. One of the drawbacks of adopting word-based embeddings that they require huge amounts of data to train and cannot be generalized to unseen (out-of-vocabulary(OOV)) words. This is evident from the pre-trained embeddings GloVe [67] which has 840 billion words and yet faces the OOV problem. One way to remedy this flaw is to use random vector values for all OOV words. However, this random assignment of word vectors "confuses" the underlying classification model. In order to remove this flaw, word embeddings models are extended to character-based embeddings [44]. All of these methods depend in some way or another on the Distributional Hypothesis, which states that words that appear in the same contexts, share semantic meanings [68, 91, 60, 67, 49]. Such representation has been more effective as compared to BoW models for preserving semantic meanings of words, sentences, and eventually documents, thus, leads to a higher performance of underlying classifiers [91, 80]. These word vectors are available in the form of pre-trained models such as word2vec [59], GloVe [67], and ELMo [68]. However, robust embedding models are limited to English language only and no such pre-trained embeddings are available publicly for Roman Urdu language. To learn the embedding vectors, deep learning techniques are employed while considering the contextual appearance of words within a fixed size window of words in a sentence or document [67]. Embeddings successfully capture the semantic information, unfortunately, it does not account for syntactic information (i.e., order of the words/characters that form a sentence or document).

2.2 Deep Learning

Performing analytics tasks on the short text is challenging due to limited information that a short text holds. Therefore, it is plausible to capture as much semantic and syntactic information as possible while performing text analytics tasks. As discussed in Section 2.1,

[1] https://www.tensorflow.org/tutorials/word2vec

word embeddings are efficient at capturing the semantic information, accounting for syntactic information however, remains a challenge. Let us consider the following two sentences: (1) *"Mary is beautiful."* (2) *"Is marry beautiful?"*. Although these two sentences are clearly in two different contexts, they would produce similar sentence representations if any traditional classifier is used that does not account for the position or order of the words in a sentence. This is because both sentences have the same words except punctuation, which is often removed as a part of preprocessing.

To this end, deep learning (DL) architectures are used to learn sentence representation and perform classification, which not only capture the word order but also learn other linguistic aspects of natural language, hence, perform better than traditional classifiers such as SVM. Deep learning has been characterized as the re-branding of artificial neural networks. Before 2006, training deep architectures were unsuccessful and challenging to train due to computational limitations. After 2006, new methods for unsupervised pre-training or feature extraction/generation have been developed. In particular, Deep Belief Networks (DBNs) [33] acted as a pivotal work to reduce the computational cost as these networks were trained for one layer at a time. In the following year, feasibility of using Restricted Boltzmann Machines (RBMs) for collaborative filtering was examined [73] and in 2008 autoencoders were proposed for unsupervised feature learning [84]. These methods are later generalized to work with document representation and learn distributed representation of text [60]. Deep learning is capable of learning multiple linguistic features automatically for various text analytics tasks, which are not possible to define manually using traditional handcrafted features. Subsequent subsections present the introduction, merits, and demerits of two specific deep learning architectures for NLP tasks called Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN).

2.2.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is proved to achieve the state of the art results in the field of image processing and computer vision, even with a limited amount of information. CNNs are a specialized set of algorithms of deep learning. Opposed to feedforward neural networks, CNNs do not learn a single global weight matrix between two layers, but instead, they aim to find a set of locally connected neurons. Their name comes from a “convolution" operator or simply “filter" that is learned during the training process and is applied to the data representation in each layer. CNN attempts to learn multiple levels of representation of increasing complexity and abstraction [48]. Although originally CNNs were designed for images but later, those were generalized to work with text representations and encoding tasks [41]. The architecture of CNN is modified to extract n-gram information from the text

to perform any NLP task [41]. It is successfully used in multilingual sentiment classification text [57, 4] and paraphrase identification of short text [29]. Although, CNNs are better at exploiting “spatial correlation", they lack the ability to retain long range information.

2.2.2 Recurrent Neural Network (RNN)

In natural language, understanding of a word is based on the understanding of previous words. The order of the words in a text matter in terms of context, semantics, and overall meaning and reasoning from previous events can be used to infer later ones. Traditional Artificial Neural Networks cannot capture this phenomenon of linguistics. Recurrent Neural Networks (RNNS) are specialized deep learning architectures that take one time steps as input at a time and output encoding for a time series data. In NLP, this translates to considering a sequence of the words in the order that they appear to model the language. This is achieved by recurring with respect to words over time. Lets consider a sequence:

$$x = [hello, world]$$

This sequence is fed to a single neuron which has a single connection to itself. At the time step 0, the word “hello" is given as input while at time step 1, “world" is given as input, and so on. At final time step, this generates a single encoding/representation of the whole sequence, which is more abundant and holds context, semantics, and syntactic information [1]. However, RNN has two major limitations. First, as the gap between two timestamps grows (i.e., sequence of words get lengthier), the older information “fades away". This phenomenon is known as the vanishing gradient problem. This occurs due to consecutive matrix multiplications when the gradient in backpropagation becomes tiny or zero. Second limitation is opposite to vanishing gradient and is known as exploding gradient where the gradient gets so big that the model cannot handle the larger values. To cater these limitations, specialized RNN architectures called Long Short-term Memory (LSTM) [34] and Gated Recurrent Unit (GRU) [11] are proposed. Although, these two architectures are similar in many ways, there are a few differences. GRU has a fewer parameters but it does not allow controlled exposure of memory content and consequently, it exposes its full content to other units in the network. While LSTM controls the amount of memory content by output gate [12]. Furthermore, LSTM is more efficient at remembering longer sequences as compared to GRU. Thus, we prefer LSTM over GRU for this study.

Long Short-term Memory (LSTM): LSTM, or Long Short term memory[34], is a special kind of RNN and can understand both long and short term dependencies and is able to connect the gap between two far-away words. As highlighted above, unlike traditional RNNs,

LSTM prevents backpropagated gradients from vanishing or exploding. Instead, errors can flow backward through unlimited numbers of virtual layers. That is, LSTM can learn tasks that require memories of events that happened hundreds or even millions of discrete time steps ago [74].

Around 2007, LSTM became popular for tasks that required sequence modeling such as speech recognition and connected handwriting recognition. Since 2014, LSTM has become a go-to RNN architecture for natural language processing tasks. LSTM can learn to recognize context-sensitive languages, which makes it a perfect candidate to model natural language.

Given these benefits, CNN and LSTM are used in a plethora of text analytics tasks such as image caption generation [85], machine translation [79], language modeling [40], and multilingual language processing [26].

2.3 Paraphrase Identification in Short Text

Automatic paraphrase detection has been widely studied in the NLP and IR communities. It has many applications in text analytics such as plagiarism detection, natural language inferencing, information retrieval, query ranking, question-answer systems, and text summarization [1]. It is a binary classification task which detects whether or not a pair of texts is semantically identical. Traditional approaches for paraphrase identification take advantage of removing stop words, followed by stemming and then representing each text as a bag of words. A similarity measure (e.g., Jaccard/Cosine similarity) is then applied, and the pair is reported to be a paraphrase if the similarity is found above a certain threshold. However, this technique yields inferior generalization in case of the short text as there can be two short sentences with exact same words but different meanings (recall example in 2.2), hence paraphrase detection based on the context/semantic cannot be achieved [88]. Another approach used by early researchers was to use hand-crafted features to capture *n*-gram overlapping, words reordering, and syntactic alignments phenomenon [32, 92]. Such an approach can work well on some specific datasets but has a pitfall that it ignores lower level interactive features between two phrases or sentences.

Short text can be clean (i.e., that follows proper grammar and formal diction like news headlines) or noisy (i.e., having informal verbiage and spelling variations like tweets). An abundance of work has been done on clean text paraphrase detection. A weighted term frequency approach for text pair representation along with *n*-gram overlap features between two texts is proposed in [38] to train SVM classifier. In [62], lexical (Parts of Speech (POS) overlap, minimum edit distance, text alignment) and semantic (Named-entity overlap, topic modeling) features between a pair of input text are used to train support vector regressor

for paraphrase identification of news tweets in Arabic language. A probabilistic model that relies upon the similarity between syntactic trees of two input documents is proposed in [13]. The authors in [64] devise a method for computing semantic similarity based on the lexical database. The model depends on WordNet meanings and syntactic roles among words in two documents. The major shortfall of this approach is that it is impractical for the noisy text as lexical databases lack evolving jargon, self-created abbreviations, and informal spellings.

Several studies have been carried out that utilize deep learning architectures for paraphrase detection in clean short text. A recursive auto-encoder for reliably understanding the context of texts and performing paraphrase detection is proposed in [75]. This architecture forms a recursive tree and performs dynamic pooling to convert the input into fixed-sized representations. However, making a tree requires parsing hence this approach is less prone to be scalable. In [35], patterns learned on a pair of text through CNN are matched at different levels of abstraction, introducing explicit interaction between the two documents during the learning process. In [19], five lexical metrics are used for reliable semantic similarity detection, where abductive networks are employed to get a composite metric that is used for classification. A method of decomposing text pair to similar and dissimilar components is proposed in [94]. A CNN is then trained to convert these components into a fixed dimension vector and classification. In [99], three attention schemes are used in CNN to form interdependent document representations. However, the sentences were matched at single level of granularity. More recently, a multi-perspective matching model (BiMPM) is proposed in [91], that uses character-based LSTM to learn word representations and a bi-directional LSTM for document representation for each text. After that, it performs four types of matching in "matching-layers" and finally, all these representations are aggregated by an additional bi-directional LSTM for paraphrase detection. One extension of this work is the neural paraphrase detection model based on a self-attended feed-forward network with pre-trained embeddings on a huge corpus of another paraphrase data [81]. It is shown that this approach outperforms BiMPM model in terms of testing accuracy with fewer parameters.

As compared to clean short text, less work has been done for noisy short text paraphrase detection. In [98], string-based features (whether the two words, their stemmed forms, and their normalized forms are the same, similar or dissimilar), common POS, and common topic (word's association with a particular topic) between a pair of text are used as features and a novel multi-instance learning paraphrase model (MultiP) is proposed. Simple lexical features based on word and character n-gram overlaps between two texts are constructed to train SVM classifier in [20]. An approach in [101] uses corpus-based (similarity between sum of word vectors of two sentences), syntactic, and sentiment polarity based features and train SVM classifier. While [100] uses ensemble approach based on seven models using word

embeddings. A major drawback of ensemble models is that it is a dataset specific combination of models, which is less probable to work on other settings. In [16], a set of lexical (character and word level *n*-grams), syntactic (words with matching POS tag, same verbs), semantic (adjective overlap), and pragmatic (subjective/objective agreement) features are identified between pair of input text, along with extensive preprocessing (spelling correction, stemming, stopwords removal, synonymous replacement). Although these features perform well on noisy short text, it is shown by the authors that it fail to get good predictive performance on clean text corpus of Microsoft Research Paraphrase (MSRP). A key problem with much of these works is that a method specifically engineered for clean text does not work for noisy text and vice versa. This calls for a single model that is oblivious to both settings.

A recent study focuses to develop a single deep learning model that performs well on both clean and noisy short text paraphrase identification [1]. A CNN and LSTM-based approach is adopted to learn sentence representations, while a feedforward network is used for classification. It also utilizes hand-crafted linguistic features, which improves paraphrase detection accuracy. Our approach follows this trend and we focus on a single model for paraphrase identification in both clean and noisy datasets. We also use linguistic features but our set of these features is different than [1]. Furthermore, these models take advantage of just one kind of feature learning, i.e., either LSTM or Convolutions. Our proposed model takes advantage of multiple feature learning aspects in a local context. As a result, the proposed approach outperforms existing approaches on three text analytics tasks.

One major downside of deep learning models is that inherently, they require a large amount of training data, and smaller datasets limit the quality of the model [16, 81]. For instance, the AskUbuntu dataset [18] contains very few annotations, thus limiting the generalization performance of the model [81]. The ability to augment the data with additional sound annotations without requiring human intervention can improve the performance of deep models [1, 81]. Such data augmentation has been shown to be fruitful for data analytics when only a piece of limited ordinal information about the pairwise distance between objects is provided [46, 45, 31]. Data augmentation has also been shown to be prolific in image classification [89]. Here, standard image processing such as cropping, rotation, and object translation is done to generate additional image samples. To the best of our knowledge, a systematic procedure for augmenting paired data without relying upon the object's content has not been presented earlier.

2.4 Text Analytics in Informal and Multilingual Short Text

Most of the existing language resources, such as datasets and language models, are limited to the English language. However, there is a sizeable body of short textual content that is written in informal multilingual text (such as tweets and SMS) [4, 58]. Roman Urdu (RU) is one of such languages. Informal and noisy text in such language pose a challenge while performing text analytics. For instance, lexical variations reduce the amount of information available for classifiers as each spelling variation is treated as a distinct word, although they are the same [70]. Literature shows that the presence of non-standard terms in training data has a severe effect on the classifier's accuracy. For example, it has been reported that the Stanford named entity recognizer (NER) experiences a performance degradation from 91% on formal text to 46% on tweets (which are written informally) [54]. Therefore, before performing analytics on informal short text, early studies normalize multiple forms of words to their root word. [28] identifies and normalizes lexical variations in the short text by building a classifier and generating correct candidates based on morphophonemic similarity while [53] uses phonetic and string similarity to normalize non-standard words obtained from Twitter and SMS into standard English terms. However, such techniques for normalization apply to standard languages for which in-vocabulary (such as English dictionary) terms are known. But for transliterated, under-resourced languages, such as RU, the aforementioned techniques are not desirable for practical considerations since standard in-vocabulary terms are not available. Similarly, the use of language morphology for normalization is not possible since the stems, root words, suffixes, or prefixes cannot be identified [2].

In Natural Language Processing (NLP), deep learning has revolutionized the modeling and understanding of human languages. The richness, expressiveness, ambiguities, and complexity of the natural language can be addressed by deep neural networks without the need to produce complex engineered features [14]. Deep learning models have been successfully used in many NLP tasks involving multilingual text. A Convolutional Neural Network (CNN) based model for sentiment classification of a multilingual dataset was proposed in [57]. However, a particular record in the dataset belonged to one language only. In our case, a record can have either one or two languages. There is very little published work on this specific setting. One approach for an under-resourced language is to adapt the resources from resource-rich language [102]. However, such an approach is not generalizable in the case of RU, as it is an informal language with no proper grammatical rules and dictionary. Upon that, multilingual text (utilization of two or more languages in the single text) acts as a catalyst to the challenges faced. Another way to improve the performance of text analytics tasks on multilingual text is to annotate each word with its respective language(code-switch indication) [56, 96]. However, it is not scalable for large

data as the annotation task becomes laborious. In an effort to handle multiple languages in a multilingual textual content, some researchers translate the text into English and then utilize the resources of the English language to perform the task in hand. This has been successfully applied to Arabic, Chines, German, French, and Japanese [93, 102, 10]. A serious limitation to this approach is that it is only applicable for languages that have a robust translation resource available, while RU lacks such resources.

Realizing these limitations, we, therefore, develop two multilingual datasets (code-switched in RU and English) and propose a deep learning model to perform text analytics (multi-class classification and sentiment classification) on these datasets. Our model works well without any translation, lexical normalization, or code-switching indication and shows a higher performance on both datasets as compared to baseline multilingual models.

Chapter 3

Multi-cascaded Model and Data Augmentation for Enhanced Paraphrase Detection

In recent years, short text in the form of posts on microblogs, question answer forums, news headlines, and tweets is being generated in abundance [8]. Performing NLP tasks is relatively easier in longer documents (e.g. news articles) than in short texts (e.g. headlines) because, in longer documents, greater context is available for semantic understanding [71]. Moreover, in many cases, short texts (e.g. tweets) tend to use informal language (spelling variations, improper grammar, slang) compared to longer documents (e.g. blogs). Thus, the techniques tailored for formal and clean text do not perform well on informal one [16], which call for a need to develop an approach that can work in both settings (i.e., clean and noisy informal text) [1].

A paraphrase of a document is another document that can be different in syntax, but that expresses the same meaning in the same language. Automatically detecting paraphrases among a set of documents has many significant applications in natural language processing (NLP) and information retrieval (IR) such as plagiarism detection [5], query ranking [24], duplicate question detection [91, 7], web searching [87], and automatic question answering [21].

Paraphrase detection is a binary classification problem in which pairs of texts are labeled as either positive (paraphrase) or negative (non-paraphrase). In this setting, pairs of texts are mapped into a fixed-dimensional feature-space, where a standard classifier is learned. Feature maps based on lexical, syntactic and semantic similarities in conjunction with SVM are proposed in [16, 20]. More recently, it has been demonstrated that for short text, deep learning-based pairs representations and classification yield better accuracy [1].

Many deep learning-based schemes employ one or two Convolutional Neural Network (CNN) or Long Short Term Memory (LSTM) based models to learn features and make predictions on clean texts [91, 35, 81], while a recent model also incorporates linguistic features to detect paraphrases in both clean and noisy short texts [1]. For many NLP tasks involving short texts, it has been shown that developing wider models can yield significant gains [72].

While deep models produce richer representations, they require large amounts of training data for a robust paraphrase detection system [16]. Thus, for small datasets, such as Microsoft Research Paraphrase (MSRP) corpus and SemEval-2015 Twitter paraphrase dataset (SemEval), handcrafted features and SVM classifier have been widely used [16, 38]. Labeling pairs of documents in a human-based computation setting (e.g. crowd-sourcing) is costly [97]. Therefore, [1] and [81] add to the training set each labeled pair also in the reversed order. However, this simple data augmentation strategy can be extended in a systematic manner by relying upon set and graph theory. For instance, consider four documents: (a) *How can I lose weight quickly?* (b) *How can I lose weight fast?* (c) *What are the ways to lose weight as soon as possible?* (d) *Will Trump win US elections?*. If in the annotated corpus, documents (a) and (b) and documents (b) and (c) are marked as paraphrases, then by transitive extension, documents (a) and (c) can also be considered as paraphrases. Similarly, if documents (a) and (b) are labeled as paraphrase, while documents (b) and (d) are labeled as non-paraphrase, then a new non-paraphrase pair based on document (a) and (d) can be inferred reliably. Such a strategy can be used to generate additional annotations in a sound and cost-effective manner, and potentially enhance the performance of deep learning models for paraphrase detection.

In this chapter, we propose a data augmentation strategy for generating additional paraphrase and non-paraphrase annotations reliably from existing annotations. We consider notions of paraphrases and non-paraphrases as binary relations over the set of documents. Representing the binary relation induced by the paraphrase labels as an undirected graph and performing transitive closure on this graph, we include additional paraphrase annotation in the training set. Similarly, by comparing paraphrase and non-paraphrase annotations we infer additional non-paraphrase annotations for inclusion in the training corpus. Our strategy involves several steps and a parameter through which the data augmentation can be tuned for enhanced paraphrase detection.

We also present a robust multi-cascaded deep learning model (McM) and show its usefulness for the task of paraphrase detection in short texts. Our model utilizes three independent CNN and LSTM (with and without soft attention) cascades for feature learning in a supervised manner. We also employ a number of additional linguistic features after

corpus-specific text preprocessing. All these features are fed into a discriminator network for final classification.

To show effectiveness of our approach we evaluate the data augmentation and deep model on three benchmark short text datasets (MSRP and Quora (clean), and SemEval (noisy)) and show that the proposed model outperforms current best performances on benchmark datasets of both types. We also perform extensive comparisons with the state-of-the-art methods. Finally, we analyze the impact of various data augmentation steps and different components of the multi-cascaded model on paraphrase detection performance.

The rest of the chapter is organized as follows: We present the theory behind data augmentation strategy and algorithm in Section 3.1. The multi-cascaded model for paraphrase detection along with detail of each cascade is presented in Section 3.2. Section 3.3 shows the details on experimental setup, results of augmentation on all three datasets, preprocessing carried out for each dataset, and linguistic features. This section also showcase an ablation study, where one cascade is ignored at a time in order to highlight the importance of multi-cascaded learning. Finally, we present and compare the results of our approach with existing state-of-the-art for all three datasets in Section 3.4.

3.1 Data Augmentation for Paraphrase Detection

We start the presentation of our enhanced approach for paraphrase detection by discussing the proposed data augmentation strategy. Paraphrase annotation is costly and time-consuming while deep learning approaches demand a large corpus of annotated paraphrases. To address this problem, we develop strategies for generating additional annotations efficiently in a sound manner. We rely upon set theory and graph theory to model the paraphrase annotation problem and present an algorithm for generating additional data for training.

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_{|\mathcal{D}|}\}$ be the set of documents in the annotated corpus. The corpus contains annotations for paraphrases and non-paraphrases. The triplet $(d_i, d_j, 1)$ indicates that documents $d_i \in \mathcal{D}$ and $d_j \in \mathcal{D}$ are considered paraphrases, and the triplet $(d_i, d_j, 0)$ denotes that documents $d_i \in \mathcal{D}$ and $d_j \in \mathcal{D}$ are considered non-paraphrases. Let N_p and N_{np} denote the numbers of paraphrase and non-paraphrase annotations in the corpus, and $N = N_p + N_{np}$ be the total number of annotations in the corpus. Note that in practice, only a fraction of the pairs of documents in $\mathcal{D}$ will be annotated in the corpus, i.e., $N \ll |\mathcal{D}|^2$.

The information contained in the annotated corpus can also be represented as a graph over the vertex set in $\mathcal{D}$. Each triplet corresponds to an edge in the graph with its label (1 or 0) indicating whether the two documents are considered paraphrases or not. For example, the triplet $(d_i, d_j, 1)$ is represented by an edge between vertex $d_i \in \mathcal{D}$ and vertex $d_j \in \mathcal{D}$ with

label 1. We assume that each edge can have a single label only, i.e., there are no conflicts in the annotated corpus whereby the same pairs of documents are labeled as both 1 and 0. If such conflicts do exist, they are removed from the corpus.

3.1.1 Generating Additional Paraphrase Annotations

The notion of pairs of documents in $\mathcal{D}$ being considered paraphrases can be captured by the notion of binary relation in set theory. Let $\mathcal{R}_p \subset= (\mathcal{D} \times \mathcal{D})$ define the binary relation over $\mathcal{D}$ such that $\forall i \forall j (d_i, d_j) \in \mathcal{R}_p$ implies that d_i is a paraphrase of d_j. In general, the following two properties hold for $\mathcal{R}_p$:

1. $\mathcal{R}_p$ is reflexive, i.e., $\forall i, (d_i, d_i) \in \mathcal{R}_p$.
2. $\mathcal{R}_p$ is symmetric, i.e., $\forall (i, j), (d_i, d_j) \in \mathcal{R}_p \implies (d_j, d_i) \in \mathcal{R}_p$.

The notion of paraphrasing is not defined precisely in linguistics. The boundary between paraphrases and non-paraphrases can lie on the continuum between (strong) paraphrases on one end and (strong) non-paraphrases on the other [83]. For clean texts (e.g., news headlines), we may approximate the notion of paraphrase by the notion of semantic duplicate, i.e., $(d_i, d_j, 1)$ implies that documents d_i and d_j are considered duplicates semantically. In set theory, this corresponds to the equivalence relation. The equivalence relation $\mathcal{R}_e$ over $\mathcal{D}$ possesses the following property in addition to properties 1 and 2 listed above:

3. $\mathcal{R}_e$ is transitive, i.e., $\forall (i \neq j \neq k), [(d_i, d_j) \in \mathcal{R}_e \wedge (d_j, d_k) \in \mathcal{R}_e] \implies (d_i, d_k) \in \mathcal{R}_e$

Transitivity can be a strong property when applied to the notion of paraphrases, especially for noisy text. Therefore, we consider $\mathcal{R}_p$ to include transitive extensions of the direct relation $\mathcal{R}$ to a pre-selected order $K \geq 1$. That is, $\mathcal{R}_p = \mathcal{R}_p \cup \mathcal{R} \cup \mathcal{R}^1 \cup \cdots \cup \mathcal{R}^K$ where $\mathcal{R}$ denotes the relation that two documents are paraphrases due to a direct relationship between them and relation $\mathcal{R}^K$ indicates that two documents are considered paraphrases because they are $K \geq 1$ intermediate documents relating them ($\mathcal{R}^1$ is the transitive extension of $\mathcal{R}$, and $\mathcal{R}^K$ is the transitive extension to order K of $\mathcal{R}$). If $K = *$ (i.e., maximum order extension is done) then we achieve a transitive closure of $\mathcal{R}$.

Now, consider the graph on the vertex set $\mathcal{D}$ induced by edges labeled with 1, i.e., pairs considered to be paraphrases in the annotated corpus. These pairs do not necessarily induce the relation $\mathcal{R}_p$ on $\mathcal{D}$. We therefore add more pairs to transform it into the desired relation. More formally:

1. For each $d_i \in \mathcal{D}$, we add $(d_i, d_i, 1)$ in the corpus, i.e., we declare each d_i a paraphrase of itself. We call this step generation by reflexivity.

2. For each $(d_i, d_j, 1)$, in the corpus, we add $(d_j, d_i, 1)$ in the corpus, i.e., we consider d_j and d_i to be paraphrases of each other. We call this step generation by paraphrase symmetry.

3. For every chain of annotations $(d_i, d_{j_1}, 1), (d_{j_1}, d_{j_2}, 1), \ldots, (d_{j_{K-1}}, d_{j_K}, 1), (d_{j_K}, d_k, 1)$ starting at d_i and ending at d_k with at most K intermediate documents in the annotated corpus, we add $(d_i, d_k, 1)$ and $(d_k, d_i, 1)$ in the corpus. Thus, we consider d_i and d_k to be paraphrases of each other if these documents are connected by at most K intermediate documents. We call this step generation by paraphrase transitive extension.

In graph terminology, step 1 corresponds to adding self-loops on each vertex with label 1 while in step 2, we ignore the direction on all edges by considering the graph as an undirected graph. Step 3 corresponds to performing a transitive extension on the undirected graph induced by edges labeled with 1. Every vertex needs to be made adjacent to all vertices that are reachable from it in $\leq K$ hops. This can be done with a single BFS (breadth first search) or DFS (depth first search) on the graph.

Note that transitive extension transforms the graph into a collection of cliques (fully connected vertices) where each clique is a paraphrase class representing a unique concept.

3.1.2 Generating Additional non-Paraphrase Annotations

Let $\mathcal{R}_{np}$ denote the binary relation that two documents in $\mathcal{D}$ are considered non-paraphrases. By definition, this relation is irreflexive, i.e., $\forall i, (d_i, d_i) \notin \mathcal{R}_{np}$. Obviously, a document cannot be a non-paraphrase of itself. The relation $\mathcal{R}_{np}$ is symmetric, i.e., $\forall i, j, (d_i, d_j) \in \mathcal{R}_{np} \implies (d_j, d_i) \in \mathcal{R}_{np}$. Transitivity does not hold for relation $\mathcal{R}_{np}$; if $(d_i, d_j) \in \mathcal{R}_{np}$ and $(d_j, d_k) \in \mathcal{R}_{np}$, then we cannot say for sure that d_i is not a paraphrase of d_k.

Additional non-paraphrase annotations can also be inferred by comparing them with paraphrase annotations. For example, if $(d_i, d_j, 0)$ and $(d_j, d_k, 1)$ (i.e., d_i and d_j are non-paraphrases, while d_j and d_k are paraphrases), then $(d_i, d_k, 0)$ must also be true (i.e., d_i and d_k are non-paraphrases). Of course, if d_i and/or d_k lie within two different paraphrase classes, then all pairs of documents in the classes will be considered non-paraphrases. These relations are also included in $\mathcal{R}_{np}$.

Now, consider the graph on vertex set $\mathcal{D}$, induced by edges labeled with 0, i.e., non-paraphrases in the annotated corpus. These pairs do not necessarily induce the relation $\mathcal{R}_{np}$ on $\mathcal{D}$. We, therefore, add more pairs to transform it into the desired relation as follows:

1. For each $(d_i, d_j, 0)$, in the corpus, we add $(d_j, d_i, 0)$ in the corpus, i.e., we consider d_j and d_i to be non-paraphrases as well. We call this step generation by non-paraphrase symmetry.

2. For every $(d_i, d_j, 0)$ in the corpus, let $\mathcal{C}$ and $\mathcal{C}'$ be the cliques (paraphrase classes) containing d_i and d_j, respectively, we add $(d_m, d_n, 0)$ and $(d_n, d_m, 0)$ for each $d_m \in \mathcal{C}$ and $d_n \in \mathcal{C}'$ to the corpus. We call this step generation by non-paraphrase transitive extension.

In terms of graph, the second step corresponds to making a complete bipartite graph between the vertex set of $\mathcal{C}$ and that of $\mathcal{C}'$.

3.1.3 Conflicts and Errors in Annotations

Using the principled strategy outlined earlier, conflicts and errors in annotations can be identified and potentially fixed. A conflict occurs when a pair of documents is found to be paraphrase and non-paraphrase either in the original annotated corpus or arises during augmentation. Based on our data augmentation strategy described above, the following conflicts can arise: (1) In the original annotated corpus, a pair of documents is labeled as both paraphrase and non-paraphrase. (2) Erroneous annotations can be generated during our data augmentation strategy in the following two cases: (a) when generating additional paraphrases by a transitive extension (b) when generating additional non-paraphrases by non-paraphrase transitive extension. A detailed analysis of conflicts and errors and their resolution is beyond the scope of this research in which we focus on data augmentation and its impact on paraphrase detection. Nonetheless, we believe that this is a fruitful area for future research.

In this work, we resolve the first type of conflict by removing the conflicting annotations and the second type of conflict by removing the conflicting non-paraphrase annotation and retaining the generated paraphrase annotation. We control the number of conflicts and errors by varying K and selecting/dropping specific generation steps and we evaluate these variations by their performances on paraphrase detection.

3.1.4 Algorithm

Algorithm 1 outlines our proposed strategy for augmenting data for enhanced paraphrase detection. The algorithm takes as input the set of documents $\mathcal{D}$, the (original) annotated corpus or dataset $\mathcal{A}$, and the parameter K (extension order) and it outputs the augmented annotated corpus or dataset $\bar{\mathcal{A}}$. After removing conflicts in the dataset (lines $4-5$), the algorithm proceeds with generating additional paraphrase annotations (lines $6-15$) followed by generating additional non-paraphrase annotations (lines $16-19$). Generating paraphrase annotations involve 3 steps i.e., P1 (lines $6-7$), P2 (lines $8-9$), and P3 (lines $10-15$),

Algorithm 1 : Data augmentation strategy

Input: $\mathcal{D}$ (documents), $\mathcal{A}$ (original corpus), K (extension order)
Output: $\bar{\mathcal{A}}$ (augmented corpus)
▷ Remove conflicting annotations
for all $[(d_i,d_j,0) \in \mathcal{A} \vee (d_j,d_i,0) \in \mathcal{A}] \wedge [(d_i,d_j,1) \in \mathcal{A} \vee (d_j,d_i,1) \in \mathcal{A}]$ **do**
    $\mathcal{A} \leftarrow \mathcal{A} \setminus \{(d_i,d_j,0),(d_j,d_i,0),(d_i,d_j,1),(d_j,d_i,1)\}$
$\bar{\mathcal{A}} \leftarrow \mathcal{A}$
▷ Paraphrase augmentation
▷ Step P1: by reflexivity
for all $d_i \in \mathcal{D}$ **do**
    $\bar{\mathcal{A}} \leftarrow \bar{\mathcal{A}} \cup (d_i,d_i,1)$
▷ Step P2: by paraphrase symmetry
for all $(d_i,d_j,1) \in \mathcal{A}$ **do**
    $\bar{\mathcal{A}} \leftarrow \bar{\mathcal{A}} \cup (d_j,d_i,1)$
▷ Step P3: by paraphrase transitive extension
$\mathcal{A} \leftarrow \bar{\mathcal{A}}$
for $(n \leftarrow 1 \rightarrow K)$ **do**
    for all $(d_i,d_{j_1},1) \in \mathcal{A} \wedge \cdots \wedge (d_{j_n},d_k,1) \in \mathcal{A}$ **do**
        if $(d_i,d_k,0) \in \bar{\mathcal{A}} \vee (d_k,d_i,0) \in \bar{\mathcal{A}}$ **then**
            $\bar{\mathcal{A}} \leftarrow \bar{\mathcal{A}} \setminus \{(d_i,d_k,0),(d_k,d_i,0)\{$
        $\bar{\mathcal{A}} \leftarrow \bar{\mathcal{A}} \cup \{(d_i,d_k,1),(d_k,d_i,1)\}$
▷ Non-paraphrase augmentation
▷ Step NP1: by non-paraphrase symmetry
for all $(d_i,d_j,0) \in \bar{\mathcal{A}}$ **do**
    $\bar{\mathcal{A}} \leftarrow \bar{\mathcal{A}} \cup (d_j,d_i,0)$
▷ Step NP2: by non-paraphrase transitive extension
for all $(d_i,d_j,0) \in \bar{\mathcal{A}} \wedge (d_j,d_k,1) \in \bar{\mathcal{A}}$ **do**
    $\bar{\mathcal{A}} \leftarrow \bar{\mathcal{A}} \cup \{(d_i,d_k,0),(d_k,d_i,0)\}$

while generating non-paraphrase annotations consists of two steps i.e., NP1 (lines 16 – 17) and NP2 (lines 18 – 19). It is worth noting that step P1 can be performed at any sequence while the other steps must follow the given sequence. In our experiments, we perform P1 after P2 and P3. Note that step P1 means additionally generated paraphrase pairs would be equal to the number of unique texts in the training data (minus number of pairs that were already part of the original annotations).

The worst-case computational complexity of the algorithm is defined by step P3. If Z is the size of the largest paraphrase class (max. clique size or max. node degree in graph terminology), then the computational complexity of the algorithm is $O(|\mathcal{D}|ZK)$. Note that in practice both Z and K will be much less than $|\mathcal{D}|$.

3.2 Multi-cascaded Deep Model for Paraphrase Detection

In this section, we present our multi-cascaded deep learning model for enhanced paraphrase detection. While deep models have been popularly used in recent years for paraphrase detection, they are typically tailored to either clean text (e.g., news headlines) or noisy text (e.g., tweets). Similarly, most employ a single model for feature learning and discrimination, and some do not utilize linguistic features in their models. In order to benefit from previous insights and to produce robust paraphrase detection for both clean and noisy texts, we propose three independent feature learners and a discriminator model that can consider both learned and linguistic features for paraphrase detection.

Figure 4.1 shows the architecture of our multi-cascaded model for paraphrase detection. We employ three feature learners that are trained to distinguish between paraphrases and non-paraphrases independently (in parallel). The features from these models (one layer before the output layer) are subsequently fed into a discriminator network together with any additional linguistic features to make the final prediction.

Our model takes as input a pair of documents d_i and d_j and outputs the label 1 if the documents are considered paraphrases and the label 0 if the documents are considered non-paraphrases. Let $d_i = < w_1^i, w_2^i, \ldots, w_{T_i}^i >$ be the sequence of words in a document. We represent each word in the sequence by an e dimensional fixed-length vector (word embedding). For a given paraphrase detection problem, we empirically decide length T of each document to use such that longer documents are truncated and shorter ones are padded with zero vectors. This decision is made to ensure compatible length vectors for all document pairs across different models. As discussed in Section 3.3.1, we select an appropriate length T for each dataset based on the distribution of lengths of documents in the dataset. We

experiment with several linguistic features (syntactic and lexical) for both clean and noisy text paraphrase detection. The details of these features are given in Section 3.3.3.

The details of the feature learners and the discriminator network are given in the following subsections.

3.2.1 Feature Learners

We employ three independent cascades to learn contextual features for paraphrase detection. Each cascade focuses on a different sequential learning model to extract features from different perspectives. Each cascade takes as input the pair of documents to be classified as paraphrase or non-paraphrase, and each is trained independently on the annotated corpus. The details of all three cascades are discussed in the following paragraphs.

The first cascade is based on CNN with soft-attention (Figure 3.2). The first layer for each input text (i.e., d_i and d_j) has 300 CNN filters with kernel size of 1. This layer learns convolutional feature map of uni-grams. Subsequently, soft-attention is applied to highlights words in documents d_i and d_j that are more important to achieve correct prediction. The result of soft-attention for document d_i is then subtracted and multiplied with that of document d_j to learn semantic contrariety. The next layer concatenates the output of CNN layer, soft-attention output, subtracted output, and multiplied output for both documents (one layer concatenates these 4 outputs for d_i and other for d_j independent of each other). This output is forwarded to another CNN layer containing 300 filters with kernel size of 2. The purpose

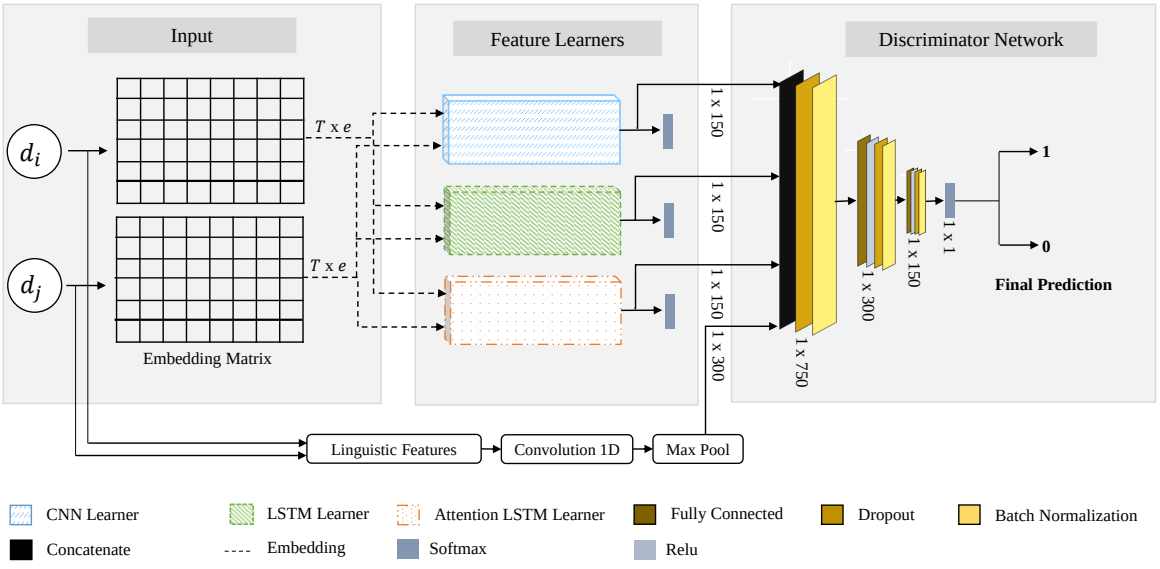


Fig. 3.1 Multi-cascaded model for enhanced paraphrase detection (Figure best seen in color)

of this CNN layer is to learn bi-gram feature representation. After this representation, global max-pooling and global average-pooling is performed to obtain most important bi-gram and average of all bi-grams respectively for both documents. These two representations are then concatenated to make a single vector of both documents (up till now, both documents were being treated separately so two streams of same functions were being produced). This outputs a vector of length 1,200. After this concatenation, a dropout and batch normalization layer is deployed to avoid any feature co-adaptation. Then this representation is forwarded to a fully connected layer with 300 units with ReLU activation followed by drop out and batch normalization. Finally, this representation is squashed into a 150-dimensional vector by another fully connected layer. This cascade is learned to detect paraphrases and non-paraphrases, and the learned 150-dimensional representation is forwarded to discriminator network for final classification.

The second cascade utilizes LSTM to encode long-term dependencies in the documents [90] (Figure 3.2). This cascade learns contextual representations without any attention or semantic contrariety. As such, it comprises of a single LSTM layer with 300 units followed

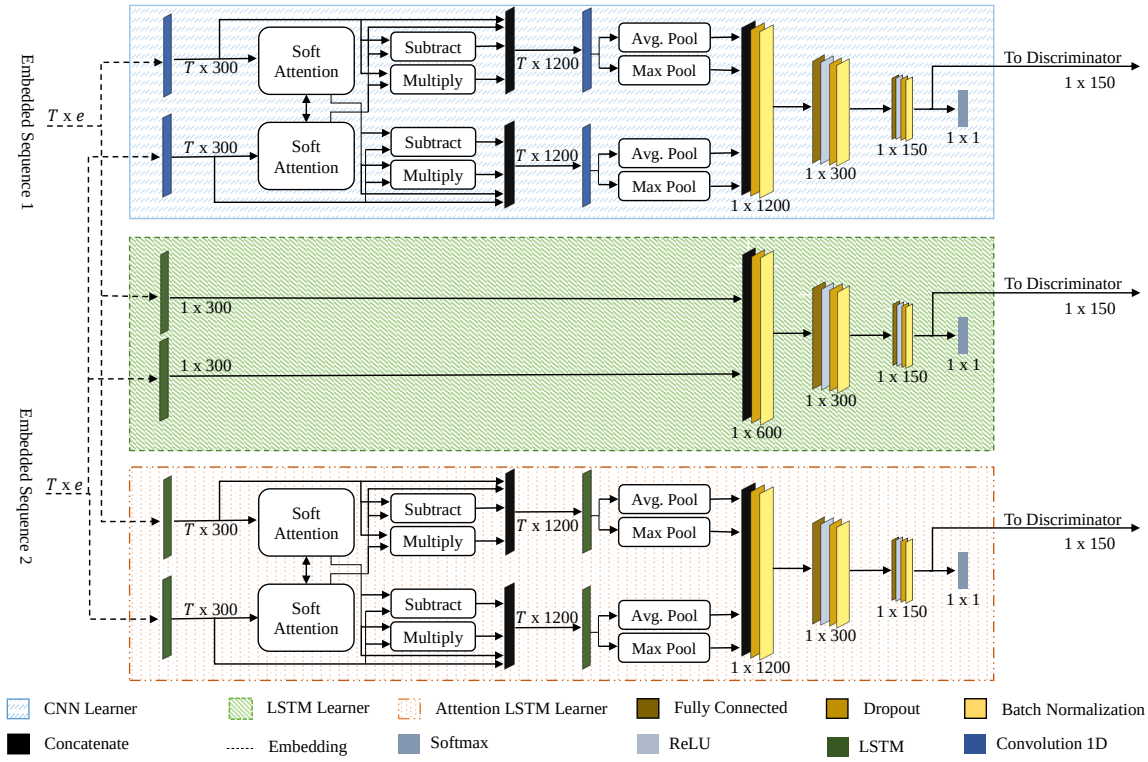


Fig. 3.2 Architecture and dimensions of output for each cascade in our multi-cascaded model (Figure best seen in color)

by fully connected hidden layers with 300 and 150 units each and an output layer. Again, the 150-dimensional representation serves as an input to the discriminator network for final classification.

The third cascade is based on LSTM with soft-attention (Figure 3.2). This cascade is similar to the first cascade but uses LSTM units instead of CNN filters. As such, it does not learn bi-gram feature representation but instead learns long-term dependencies with important sequences highlighted by attention mechanism. The 150-dimensional vector obtained before the output layer is used as an input to the discriminator network for final classification.

We empirically decide to use 300 CNN filters, 300 LSTM units, 300 units for first fully-connected layer, and 150 units for second fully-connected layer in all cascades. In all feature learning cascades, weights of CNN filters as well as weights of LSTM layers are shared among d_i and d_j. Weight sharing reduces the number of parameters required and both documents are converted to deep representations in same embedding space. We use ReLU activation for every layer except last prediction layer, which uses the softmax function. We train the model using categorical cross entropy loss. Dropout and batch normalization is used after every fully-connected layer.

3.2.2 Discriminator Network

Figure 4.1 shows the discriminator network utilized to make the final prediction. For a given pair of documents d_i and d_j, this network takes as input the features from all three cascades plus any linguistic features with dropout and batch normalization. Two fully-connected hidden layers are then used with 300 and 150 units, respectively, where each is followed by dropout and batch normalization layer. The activation function for both fully-connected layer is set to ReLU. On final layer softmax activation function is used with categorical cross-entropy as loss. This network outputs final prediction for documents d_i and d_j as either paraphrase or non-paraphrase. We also do early-stopping if validation accuracy of discriminator is not improved for 10 epochs. A checkpoint of the model is created after the training epoch at which validation accuracy is improved.

3.3 Experimental Setup

In this section, we describe the settings for the experimental evaluation of our enhanced paraphrase detection model. We discuss the key characteristics of the three datasets used in our evaluations before and after augmentation. We also present the parameter settings of the model, the text preprocessing performed, and the linguistic features used in our experiments.

Table 3.1 Statistics for all datasets

Dataset	Split	Total Pairs	Paraphrase Pairs	Non-paraphrase Pairs	Debatable Pairs
Quora	Train	384,348	139,306	245,042	–
	Dev	10,000	5,000	5,000	–
	Test	10,000	5,000	5,000	–
MSRP	Train	4,076	2,753	1,323	–
	Test	1,725	1,147	578	–
SemEval	Train	13,063	3,996	7,534	1,533
	Dev	4,727	1,470	2,672	585
	Test	972	175	663	134

3.3.1 Datasets and their Augmentation

We use three real-world datasets in our experimental evaluations. The Quora questions pairs dataset (Quora) [91] contains pairs of questions in English with their annotations (paraphrase or non-paraphrase)[1]. This dataset is collected from questions posted on the Quora question-answering website. This is a clean text dataset. The Microsoft Research Paraphrase Corpus (MSRP) [17] contains pairs of sentences in English together with their paraphrase/non-paraphrase annotation[2]. The sentences are extracted from news articles on the web. Thus, this is another example of a clean text dataset. The SemEval-2015 Twitter paraphrase dataset (SemEval) [98] contains pairs of tweets in English with their paraphrase/non-paraphrase annotation [3]. This is a noisy text dataset. The Quora and SemEval datasets have pre-defined training, development, and test sets available. The MSRP dataset has predefined training and test sets only. The statistics for each split of all the datasets are presented in Table 5.1. We describe the characteristics of the *training* set of each dataset before and after augmentation in the following subsections.

Quora Dataset

The Quora dataset contains annotations for 517,968 unique questions. We notice that there are 30 incorrect non-paraphrase annotations in the dataset. The texts in these 30 annotations are identical but they are marked as non-paraphrases. We remove these annotations from the dataset.

[1] https://github.com/zhiguowang/BiMPM
[2] https://www.microsoft.com/en-us/download/details.aspx?id=52398
[3] https://github.com/cocoxu/SemEval-PIT2015

Table 3.2 Numbers of paraphrase and non-paraphrase annotations in the datasets before and after augmentation

Dataset	Paraphrases				Non-Paraphrases		
	Original	P2	P3	P1	Original	NP1	NP2
Quora	139,306	278,612	447,378	964,509	245,042	490,015	655,219
MSRP	2,753	5,506	5,874	13,688	1,323	2,646	2,647
SemEval	3,996	7,992	94,722	107,953	7,534	15,068	23,205

An analysis of this dataset reveals that questions have an average length of about 13 words with a standard deviation of 6.7 words and a maximum length of 272 words. Based on this analysis, we select $T = 40$ for this dataset.

Table 3.2 shows the numbers of paraphrases and non-paraphrases in the Quora dataset before and after each step of augmentation. For this clean text dataset, we perform step P3 (paraphrase generation by transitive extension) using $K = *$, i.e., we generate by transitive closure. Applying step P2 (paraphrase generation by symmetry) almost doubles the number of paraphrase annotations. Performing transitive closure (step P3) generates 168,766 additional paraphrase annotations. Subsequently, generating paraphrase annotations by reflexivity (step P1) produces an additional 517,131 paraphrase pairs, bringing the total of paraphrase annotations to 964,509. As an example of generated paraphrase pairs through transitive closure, consider following three questions. (a) *What causes deja vu ?* (b) *Read below, what causes deja vu ?* (c) *What is the cause of someone frequently experiencing deja vu ?*. The questions (a) and (b) and questions (b) and (c) are marked as paraphrases in the original annotations. After transitive closure, a new pair based on questions (a) and (c) is correctly inferred.

For augmentation of non-paraphrase annotations, step NP1 (non-paraphrase generation by symmetry) brings the total number of non-paraphrase pairs to 490,015. Subsequently, performing step NP2 (generation by non-paraphrase transitive extension) an additional 165,204 non-paraphrase pairs are produced, bringing the total number of non-paraphrase annotations to 655,219. Similar to the example of paraphrase annotation generation, the proposed data augmentation strategy successfully generates non-paraphrase annotations also. Consider following four questions. (a) *Why does Quora censor opinions and answers ?* (b) *Is Quora censored ?* (c) *Why does Quora want to censor/collapse the truth ?* (d) *Why does Quora have a character limit for question titles and details ?*. In original annotations, questions (a), (b), and (c) are recorded as paraphrases while questions (b) and (d) are recorded as non-paraphrases. Using non-paraphrase transitive extension, additional non-paraphrase pairs (a), (d) and (c), (d) are correctly inferred.

Table 3.3 Examples of errors detected during paraphrase augmentation using transitive extension on Quora dataset (1 = paraphrase, 0 = non-paraphrase)

Question Pair	Original Label	Generated Label
What is the colour of the Sun? What is the color of the sun?	0	1
Is pro wrestling fake? Wwe is real fight?	0	1
How can I get free iTunes gift cards online? What 's the best way to legally get free iTunes gift cards?	0	1

After performing step P3 (generation by transitive closure), it is observed that there are $57,119$ unique paraphrase classes (cliques) corresponding to distinct concepts about which questions are being asked. Note that transitive closure converts the graph on paraphrase annotations into a disjoint collection of cliques. Therefore, the node degree after transitive closure is one less than the number of questions (nodes) in which that node lies.

As discussed in Section 3.1.3, our data augmentation strategy can highlight conflicts and errors in the original annotations. For example, a paraphrase annotation generated by transitive closure can be in conflict with an existing non-paraphrase annotation. Usually, a generated annotation is based on strong evidence of related annotations, hence, such conflicts may indicate an error in the existing annotations. We find 214 such conflicts in the training set of the Quora dataset. The number of conflicts can also be considered as a measure of annotation quality. Upon subjective evaluation of these conflicts, it is observed that annotations generated by the proposed data augmentation strategy are correct while the original annotations are erroneous. Table 3.3 shows some examples of such conflicts in the Quora dataset. If we consider the first example, the only difference between both question is the spelling of the word “color", yet it is marked as non-paraphrase in the original label while the paraphrase transitive extension marks it as paraphrase, which is correct. Similarly, in second example, same question is asked with different wordings but the original annotations mark it as non-paraphrase while the proposed data augmentation strategy correctly marks it as paraphrase. These results confirm that our data augmentation strategy works well to detect and distinguish unique concepts and generate new pairs along with their associated labels from existing annotations reliably.

MSRP Dataset

The MSRP dataset is a much smaller dataset containing annotations for $7,814$ unique sentences. There are no conflicting annotations found in this dataset. We select $T = 25$ based

Table 3.4 Numbers of paraphrase and non-paraphrase annotations after step P3 and step NP2, respectively, in SemEval dataset with different orders of transitive extension

Tr. extension	Paraphrases		Non-Paraphrases
	P3	P1	NP2
$K = 1$	30,738	43,969	18,539
$K = 2$	59,538	72,769	23,490
$K = 3$	90,042	103,273	24,175
$K = *$	94,722	107,952	23,205

on exploratory analysis of the dataset which reveals that the average length of sentences is about 19 words with a standard deviation of 5.1 word and a maximum length of 31 words.

Table 3.2 gives the numbers of paraphrases and non-paraphrases before and after augmentation in this dataset. There are only 4,076 annotations in the original train split of the dataset out of which 2,753 are for paraphrases and 1,323 are for non-paraphrases. The number of paraphrases is doubled after performing step P2 (generation by paraphrase symmetry), an additional 368 paraphrase annotations are generated in step P3 (generation by transitive closure), and step P1 generates an additional 7,814 paraphrase annotations. Step NP1 doubles the number of non-paraphrase annotations while step NP2 generates just one more non-paraphrase annotation. As seen from these results, transitive closure does not add many annotations implying that there are generally small paraphrase classes in this dataset.

In this dataset, there were no conflicts detected while employing the proposed augmentation strategy.

SemEval Dataset

The SemEval dataset has paraphrase and non-paraphrase annotations for 13,231 unique tweets. The original train split of the dataset has 3,996 paraphrase, 7,534 non-paraphrase, and 1,533 *debatable* annotations [97]. All existing studies on this dataset ignore the debatable annotations while reporting their results; therefore, we also choose to ignore these annotations in our experiments. No conflicting annotations are found in the dataset.

For this dataset, we choose $T = 19$. This number corresponds to the maximum length of tweets in the dataset with the average length being around 8 words.

Table 3.2 presents the numbers of paraphrases and non-paraphrases in this dataset before and after each step of our data augmentation strategy. These numbers are obtained when transitive closure is performed during step P3, i.e., $K = *$. It is seen that the number of paraphrases jumps from 3,996 to 107,953 while the number of non-paraphrases increases

from 7,534 to 23,205. A major increase in paraphrases occurs during step P3. This can be attributed to the following two reasons: 1) The SemEval is based on a dataset developed for the task of semantic similarity estimation. As such, the notions of paraphrase and non-paraphrase are not well separated which is then blurred by the process of transitive closure. 2) The SemEval dataset contains short and noisy text that makes annotation difficult and prone to errors. For such datasets, transitive closure can be a “blunt instrument" during the data augmentation strategy.

Table 3.4 shows the numbers of paraphrases and non-paraphrases after steps P3, P1, and NP2 when transitive extension of orders $K = 1$, $K = 2$, $K = 3$, and $K = *$ (transitive closure) is performed. The numbers for step P2 and NP1 are not shown in this table as they are identical to those given in Table 3.2. It is clear from this table that order K controls the number of paraphrases that are generated during step P3. For example, when $K = 1$ the number of paraphrases after step P3 is 30,738 which is significantly lower than 94,722 when $K = *$. Note that the number of non-paraphrases actually increases slightly as the order of transitive extension is reduced. This is due to the fact that more conflicts arise as K is increased that are resolved by retaining the generated paraphrase annotation and discarding the conflicting non-paraphrase annotation, and fewer non-paraphrase annotations will produce fewer additional non-paraphrase annotations during non-paraphrase extension (step NP2). The number of conflicts during step P3 are 4, 14, 22, and 23 for $K = 1$, $K = 2$, $K = 3$, and $K = *$, respectively.

3.3.2 Preprocessing

Text preprocessing is an essential component of many NLP applications. However, in case of short text, common text preprocessing steps such as removing punctuations and stopwords can result in loss of information critical to the application [20]. Therefore, we keep preprocessing to a minimum in our experiments. For SemEval dataset, which represents noisy texts, we perform lemmatization and correct commonly misspelled words such as *dnt* to *do not*. We use a predefined dictionary to map misspelled words to their standard forms. Preprocessed tweets are used while training our multi-cascaded model as well as to extract linguistic features. For the other two datasets, which represent clean texts, we perform preprocessing (stopword removal and lemmatization) only for computing linguistic features while the raw text is used in our multi-cascaded model. The details of the linguistic features are given in the next subsection.

3.3.3 Linguistic Features

We employ a set of NLP/linguistic features in our experiments as it has been shown that including linguistic features for paraphrase identification in short text can improve the performance of deep learning models [1]. We identify the following linguistic and statistical features to be used alongside learned features in our multi-cascaded model.

1. 2 features based on cosine similarity between TF-IDF vectors of documents d_i and d_j, before and after removing stopwords and doing lemmatization.
2. 4 n-gram overlapping ratio features based on unigrams and bigrams that are common to a given document pair, divided by total number of n-grams in d_i and d_j respectively.
3. 2 features based on cosine similarity between ELMo [68] embeddings vectors of d_i with d_j, before and after removing stopwords and doing lemmatization.
4. 2 features based on cosine similarity between Universal Encoder [4] vectors of d_i and d_j, before and after removing stopwords and doing lemmatization.
5. 1 feature based on count of unigrams that has same POS tag in d_i and d_j.
6. 6 features based on length of intersection of character bigrams, trigrams and quadgrams of d_i in d_j, before and after removing stopwords and doing lemmatization.
7. 2 features based on longest substring match in d_i and d_j, before and after removing stopwords and doing lemmatization.
8. 2 features based on longest subsequence match in d_i and d_j, before and after removing stopwords and doing lemmatization.
9. 3 syntactic features based on number of Verbs, Nouns and Adjectives common in d_i and d_j.
10. 2 Named-entity Recognition (NER) features in d_i and d_j based on number of same NER tags and numbers of same word-NER tuple.

We use all linguistic features for clean text datasets and linguistic features $1-4$ only for the noisy text dataset. Linguistic features are passed through a single CNN layer with 300 dimensions and then provided as input to discriminator network.

[4] https://tfhub.dev/google/universal-sentence-encoder/2

3.3.4 Hyper-parameters Tuning

A number of hyper-parameters are required in our multi-cascaded model for paraphrase detection. We tune and select these hyper-parameters on the development sets of each dataset using a grid search. As MSRP dataset has train and test splits only, we hold-out 10% of the training set as the development set for the purpose of hyper-parameters tuning. We decide the type of word embeddings among GloVe [5] [67], Word2Vec [60], and ELMo. For selecting an optimizer, we decide among nAdam, Adam, Adadelta, and SGD. We consider dropout rates 0.1, 0.2, 0.3, 0.4 and 0.5 in our model.

3.3.5 Performance Evaluation and Comparison

Since paraphrase detection is a binary classification problem, standard measures of performance can be used for evaluation. We report performances as percent accuracy, precision, recall, and F1-value on the test sets after training over the respective training set of the datasets. Each dataset has a fixed training and test sets. Therefore, results can be compared with previously reported results on the same datasets.

3.3.6 Implementation

We use networkX library in Python for graph analysis and data augmentation[6]. For implementing and evaluating our multi-cascaded model we use Keras[7] as the front-end with TensorFlow[8] on the backside. All model parameters or weights are initialized randomly, and to ensure reproducibility the random seed is fixed. The code and implementation setting used in our experimental evaluation is available from the website[9].

3.3.7 Ablation Study

In this section, we discuss the results for the ablation study that is performed to examine the need for multiple cascades. It is well-settled that learning in DL models is hindered by smaller datasets [16]. The MSRP and SemEval datasets only have $4,076$ and $13,063$ annotations respectively. Thus, for this ablation study, we use the non-augmented (original) version of the Quora dataset. The intuition behind choosing this particular dataset is that it is a large-scale dataset (having $384,348$ records in train split and $10,000$ records in test

[5] https://nlp.stanford.edu/projects/glove/
[6] https://networkx.github.io/
[7] https://keras.io/
[8] https://www.tensorflow.org/
[9] https://github.com/haroonshakeel/multi-cascaded-deep-network-for-paraphrase-identification

split); hence the model would not be hampered by the size of the data and findings would reflect true insights about the learning capabilities of the cascades. We design the ablation experiments as follows.

First, we only use one cascade at a time amongst all three and note down the results on the test split. Second, we make mutually exclusive pairs of cascades (i.e., CNN + Attention LSTM, CNN + LSTM, and Attention LSTM + LSTM). Finally, we use all three cascades for training and testing the model. We use hyperparameters as described in Section 3.4.1 and the random seed is fixed to mitigate the effect of randomness.

Figure 3.3 illustrates the result obtained for each configuration of ablation study. With regards to the configuration that utilizes only a single cascade, it is observed that LSTM-based cascade (learner) yields the poorest performance with an accuracy of 86.23, closely followed by CNN-based learner with an accuracy of 86.98. Most exciting results are achieved by Attention LSTM cascade, which gives an accuracy of 88.46. These results make intuitive sense as LSTM only captures the order of words and long-term dependencies of text while CNN captures *n*-gram features only. Introducing the attention mechanism generates more informative features as it learns semantic contrariety amongst the most important words between two inputs concerning the label.

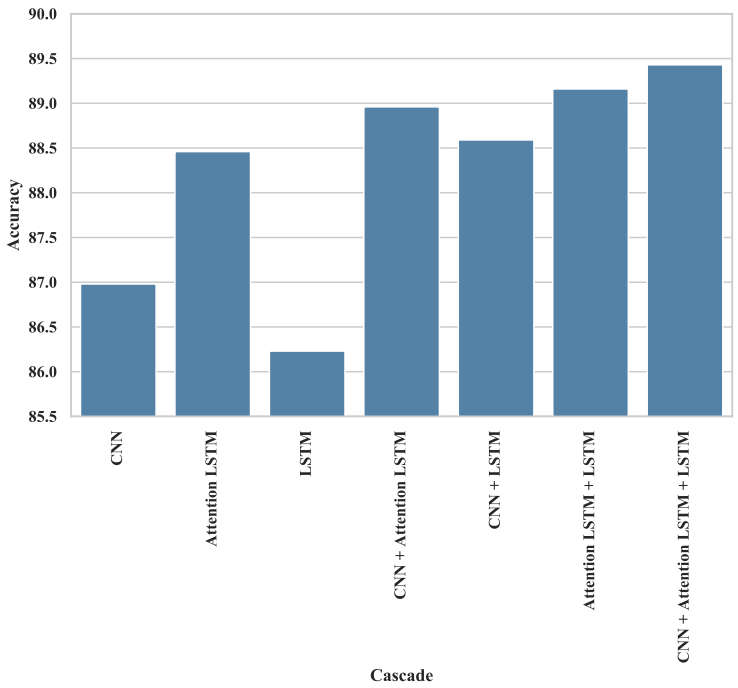


Fig. 3.3 Model accuracy for each configuration of ablation study

Next, we combine two cascades and re-run the experiments. In this particular case of pairs of cascades, CNN + LSTM gives the minimum performance with an accuracy of 88.59. This, however, is still better than the highest performing single cascade of Attention LSTM. The CNN + Attention LSTM combination gives an accuracy of 88.96 while Attention LSTM + LSTM combination shows a marginal improvement with an accuracy of 89.16. These results are insightful in a sense that Attention LSTM contributes more towards the correct prediction as compared to other cascades. However, when we combine all three cascades, the final accuracy is pushed a little further, and the model is able to achieve the testing accuracy of 89.43. Note that as the model accuracy come closer to 100, even a marginal improvement is considered significant. This is evident from the existing studies on the dataset and the incremental improvements quoted in Table 3.6.

These results lead us to conclude that *n*-gram features are more informative than capturing the order of the words (recall LSTM learner). However, when combined, they surpass single-perspective learning. Moreover, attention-based LSTM exhibits the highest performance. However, all three kinds of information are pertinent to linguistics. Thus, combining all three gives the highest predictive performance and sets justification for utilizing all three cascades.

3.4 Results and Discussion

In this section, we present and discuss the evaluation of our multi-cascaded model and data augmentations strategy in terms of paraphrase detection predictive performance. We first present results on each dataset with and without data augmentation and linguistic features. Subsequently, we discuss the performance of different components of our multi-cascaded model. Finally, we present the key takeaways from our experimental study.

3.4.1 Quora Dataset

For this dataset, the hyper-parameters of our model are tuned on the provided dev set. GloVe embeddings are selected as the best choice while among optimizers, nAdam with learning rate of 0.002 is found to be optimal. Similarly, the dropout rate of 0.1 is found to be optimal.

Table 3.5 shows the predictive performance of our enhanced paraphrase detection model on the Quora dataset. Performances (accuracy, precision, recall, and F1-score) on test sets are given for different data augmentation steps with learned features and learned plus linguistic features. We first discuss results obtained by using learned features only. Using the original data without any augmentation, our model achieves an accuracy of 89.4%. Augmenting the data with step P2 and NP1 (paraphrase and non-paraphrase generation by symmetry)

increases the accuracy slightly. Note that this augmentation step has also been performed in earlier works [81, 1]. However, noticeable increase in accuracy is observed when the data is augmented with additional paraphrases using step P2, step P3 (generation by transitive closure), and step P1 (generation by reflexivity), jumping the accuracy to 90.2%. We obtain the best performance of 90.3% when in addition to augmenting paraphrases via steps P3, P2, and P1 additional non-paraphrases are generated via step NP1. This is also the current state-of-the-art performance on this dataset.

Note that by including additional non-paraphrase annotations using step NP2 (generation by non-paraphrase transitive extension) decreases the accuracy to 89.9% from the high of 90.3% obtained when steps P2, P3, P1, and NP1 are executed. The reason behind this decrease can be determined by analyzing paraphrase concepts (clique in paraphrase graph). Recall from Section 3.1.2 that even a single edge with label 0 (a non-paraphrase annotation) between two cliques will generate a complete set of edges between the nodes of the two cliques with label 0. Thus, any error in such a non-paraphrase annotation gets magnified during the NP2 non-paraphrase augmentation step and degrades the quality of the dataset for paraphrase detection. For example, the incorrect annotation of questions *Is there a way to hack Facebook account ?* and *How can I hack Facebook ?* as non-paraphrase generates numerous erroneous non-paraphrase annotations between paraphrases of the first and second question. Therefore, step NP2 has the potential to degrade paraphrase detection performance when errors exist in non-paraphrase annotations that link large paraphrase concepts.

When we perform experiments by including linguistic features with learned features, slightly lower performances are obtained. This highlights that when sufficiently large dataset is available, deep learning models can effectively capture the semantics and contexts of short texts for improved paraphrase detection; for such datasets, the extra effort of including linguistic features is not beneficial.

Table 3.6 presents the performance of previously published work on this dataset. Accuracy values are given in this table because previous works report accuracies only. In [50], 7

Table 3.5 Paraphrase detection performance on Quora dataset

Augmentation	Learned Features				Learned + Linguistic Features			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
None	89.4	90.3	88.4	89.3	89.8	90.0	89.6	89.8
P2, NP1	89.6	90.6	88.4	89.5	74.9	67.6	**95.5**	79.2
P2, P3	89.9	90.0	89.7	89.9	**90.1**	**91.2**	88.7	89.9
P2, P3, P1	90.2	90.3	**90.1**	**90.2**	90.0	90.2	89.6	89.9
P2, P3, NP1	90.0	90.5	89.5	90.0	90.0	89.9	90.1	90.0
P2, P3, P1, NP1	**90.3**	**90.9**	89.5	**90.2**	**90.1**	89.7	90.5	**90.1**
P2, P3, P1,NP1, NP2	89.9	90.7	89.0	89.8	89.9	90.2	89.5	89.9

Table 3.6 Comparison of our model's performance with previously published performances on Quora dataset

Model	Accuracy
Wang et al. (2017) (Saimese-CNN) [91]	79.6
Wang et al. (2017) (Multi-Perspective-CNN) [91]	81.4
Wang et al. (2017) (Saimese-LSTM) [91]	82.58
Wang et al. (2017) (Multi-Perspective-LSTM) [91]	83.2
Wang et al. (2017) (L.D.C) [91]	85.6
Wang et al. (2017) (BiMPM) [91]	88.2
Tomar et al. (2017) (pt-DECATT$_{word}$) [81]	87.5
Tomar et al. (2017) (pt-DECATT$_{char}$) [81]	88.4
Lan and Xu (2018) (SSE) [50]	87.8
Our model	**90.3**

different models are re-implemented on several tasks involving sentence pairs. Quora dataset is used to get results for paraphrase detection task. We only include results of best performing model among all 7. They find that Shortcut-Stacked Sentence Encoder Model (SSE) [63] performs the best, giving testing accuracy of 87.8%. The previous best accuracy is 88.4% [81]. Our multi-cascaded model beats this result without any data augmentation with an accuracy of 89.4%. As seen from the table, our enhanced model outperforms all previous results. Ensemble BiMPM model achieves an accuracy of 88.2% accuracy while pt-DECATT$_{char}$ shows a slightly better performance with an accuracy of 88.4%. In comparison, our model achieves an accuracy of 90.3%, which is almost 2% improvement over the previous best performance. In contrast to [91], we avoid using ensemble model approach (which is computationally costly). Similarly, contrary to results in [81], our results are based on word features only and do not use computationally expensive character-based features.

3.4.2 MSRP Dataset

We use the pre-defined split provided in MSRP dataset for training and testing our model. No dev set is provided with this dataset; hence, we hold-out 10% of the training split randomly as dev set. By using a grid search, we find optimal hyper-parameters on this dev set. ELMo embeddings are found to be better for this dataset, while Adam optimizer is selected as optimal one with learning rate of 0.002. When optimizing dropout rate on this dataset, 0.5 is found to yield the best results.

Table 3.7 shows the predictive performance of our enhanced paraphrase detection model on MSRP dataset. Performances are given for configurations with and without linguistic

Table 3.7 Paraphrase detection performance on MSRP dataset

Augmentation	Learned Features				Learned + Linguistic Features			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
None	74.1	75.3	90.9	82.4	74.7	76.2	90.2	82.6
P2, NP1	74.8	76.9	88.9	82.5	75.2	77.1	89.2	82.7
P2, P3	74.4	76.0	89.9	82.3	75.0	76.8	89.5	82.7
P2, P3, P1	76.8	77.1	92.7	84.2	77.4	77.0	91.7	84.2
P2, P3, NP1	74.4	75.6	90.7	82.5	74.1	74.7	**92.5**	82.6
P2, P3, P1, NP1	**77.0**	**77.3**	**92.7**	**84.3**	**78.3**	**79.3**	91.0	**84.8**
P2, P3, P1, NP1, NP2	**77.0**	**77.3**	**92.7**	**84.3**	**78.3**	**79.3**	91.0	**84.8**

features after applying various data augmentation steps. It is observed that without data augmentation, we achieve an F1-score of 82.4%. Doubling the data (step P2 and NP1) increases performance slightly in terms of F1-score but the significant gain is obtained when augmenting the data using symmetry, transitivity, and reflexivity (steps P2, P3, P1). This configuration of augmentation gives F1-score of 84.2%. The highest F1-score in experiments without any linguistic features of 84.3% is achieved when P2, P3, P1, and NP1 augmentation steps are performed. Please note that adding a pair generated by NP2 augmentation did not affect the performance of the model as it only produces 1 additional pair (recall Section 3.3.1).

Using linguistic features along with learned features boosts the performance of the model. In comparison with experiments without linguistic features, this gain is consistent with the exception of P2, P3 and P1 augmentation scheme, where the F1-score for experiments without and with linguistic features remains the same. Without any augmentation, the F1-score is increased from 82.4% to 82.6% by introducing linguistic features, while with P2 and NP1 augmentation scheme, it is improved from 82.5 to 82.7. We achieve maximum performance using data augmentation schemes of P2, P3, P1, and NP1 while using linguistic features. This configuration yields an accuracy of 78.3% and an F1-Score of 84.8%, which is the highest among all of our experiments. Note that this augmentation scheme also has the highest F1-score when no linguistic features were used. Adding one additional pair generated by NP2 augmentation did not affect the model's performance.

These results prove that the usefulness of linguistic features is dataset domain and size-specific, and is not generalizable. In particular, the impact of linguistic features is limited when the dataset is large (e.g., Quora dataset) while it is more significant for small datasets (e.g., MSRP dataset).

We compare our best performing variation of experiments with existing state of the art approaches on MSRP dataset in Table 3.8. The current state of the art on MSRP is reported to be in [38]. They report their best results as 80.4% accuracy and an F1-score of 85.9%.

Table 3.8 Comparison of our model's performance with previously published performances on MSRP dataset

Model	Accuracy	F1-score
Socher et al. (2011) [75]	76.8	83.6
Madnani et al. (2012) [55]	77.4	84.1
Ji and Eisenstein (inductive) (2013) [38]	77.8	84.3
Hu et al. ARC-I (2014) [35]	69.6	80.3
Hu et al. ARC-II (2014) [35]	69.9	80.9
El-Alfy et al. (2015) [19]	73.9	81.2
Kenter and de Rijke (2015) [43]	76.6	83.9
Eyecioglu and Keller (2015) [20]	74.4	82.2
He et al. (2015) [29]	78.6	84.7
Dey et al. (2016) [16]	–	82.5
Wang, Mi et al. (2016) [94]	78.4	84.7
Yin et al. (2016) [99]	78.9	**84.8**
Pagliardini et al. (2018) [65]	76.4	83.4
Ferreira et al. (2018) [23]	74.08	83.1
Agarwal et al. (2018) [1]	77.7	84.5
Arora and Kansal (2019) [3]	**79.0**	–
Our model	78.3	**84.8**

However, they assume that they have access to testing data at the time of training a model and call it a form of transductive learning. On the other hand, our model is based on inductive learning where training is done in total isolation from the test split. Therefore, it is only fair to compare both models in inductive setup. The training of the model by [38] in inductive setup yields 77.8% accuracy and F1-score of 84.3% [1]. These results are far less than what were originally reported in [38] using transductive learning. Thus, the current state-of-the-art results in inductive setup are reported by [29] and [94], with an F1-score of 84.7%. Our best model yields an F1-score of 84.8%, which is slightly higher than previous state-of-the-art.

In terms of accuracy, the highest performance is reported in [3] which is 79.0%. However, the F1-score was not reported by the authors. As the class label distribution is highly skewed in this dataset (recall Table 5.1), the accuracy is not a good measure of performance. In such cases, it is plausible to use F1-score to measure the predictive performance of the models [76]. Therefore, despite higher accuracy reported by the authors, it cannot be concluded that their model yields higher performance than other reported results.

3.4.3 SemEval Dataset

In SemEval dataset, the provided dev split is used to fine-tune hyper-parameters using grid search. We find that ELMo embeddings yield the best results when Adam is used as optimizer with learning rate 0.002. The dropout rate is found to be 0.2.

Table 3.9 presents the predictive performance of our model on this dataset. In this table, data augmentation is done with full transitive closure, i.e., $K = *$ for step P3. Without data augmentation and linguistic features, our model achieves an F1-score of 40.2%. However, when we apply data augmentation, the F1-score tends to increase. The maximum F1-score without linguistic features is 64.2% which is achieved with P2, P3, P1, and NP1 augmentation. This is consistent with the results of other two datasets used in the study, which also yield

Table 3.9 Paraphrase detection performance on SemEval dataset

Augmentation	Learned Features				Learned + Linguistic Features			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
None	51.1	27.1	78.9	40.2	**89.0**	70.8	**80.6**	**75.4**
P2, NP1	78.8	48.5	21.1	29.4	88.9	**82.5**	59.4	69.1
P2, P3	82.5	57.7	60.0	58.9	84.6	63.7	61.1	62.4
P2, P3, P1	82.2	56.1	68.6	61.7	87.6	77.5	57.1	65.8
P2, P3, NP1	**84.4**	**62.6**	62.3	62.5	84.0	60.5	67.4	63.8
P2, P3, P1, NP1	84.1	60.7	68.0	**64.2**	84.4	60.0	76.0	67.0
P2, P3, P1, NP1, NP2	82.7	57.1	**68.6**	62.3	84.7	62.4	67.4	64.8

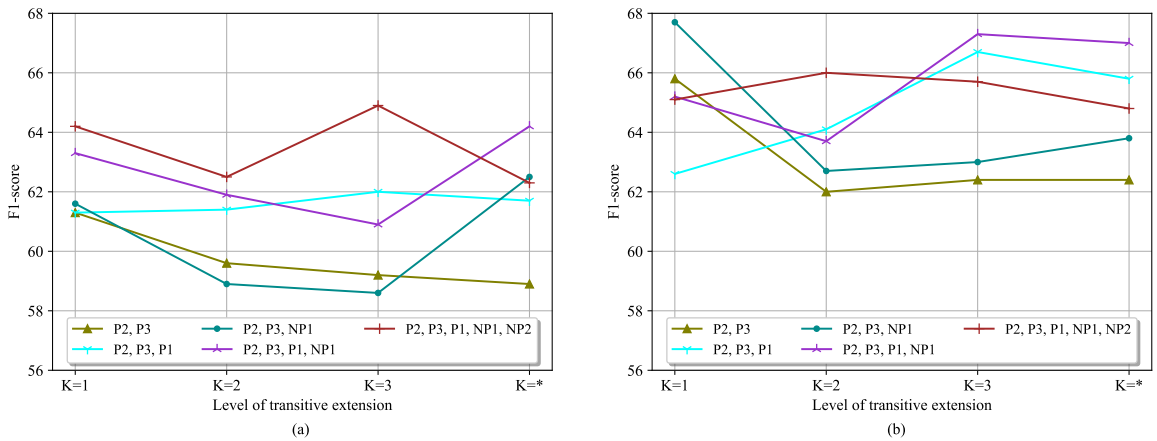


Fig. 3.4 Performance of our model with paraphrase transitive extensions of order $K = 1$,$K = 2$,$K = 3$, and $K = *$ on SemEval dataset; (a) learned features, (b) learned + linguistic features

maximum performance on this particular augmentation combination. In noisy text, linguistic features affect performance by a large margin.

Our enhanced model outperforms existing approaches on SemEval dataset in terms of F1-score when no augmentation is done but linguistic features are provided along with learned features. This particular experiment yields Precision and Recall of 70.8% and 80.6%, respectively. However, it is noticed that precision is lower than recall. When we augment the data using P2 and P3, the precision and recall values tend to become closer, highlighting that model is more robust in distinguishing the two classes. But, the F1-score drops significantly.

The reason behind this degradation in performance can be found by investigating pairs generated by transitive closure. It is observed that as we move along the path in graph to find transitive pairs, the meaning of text tends to change, and it becomes more probable that the two documents are no longer paraphrases. This phenomenon is more likely in a noisy short text such as the SemEval dataset. Therefore, instead of applying transitive closure with $K = *$, transitive extension to an order K is plausible. To investigate the effect of K in transitive extension, we perform same experiments with $K = 1$, $K = 2$, and $K = 3$. The results of these experiments in terms of F1-score is given in Figure 3.4 (plots (a) and (b) are for without and with linguistic features, respectively). Complete tables of results are included in Appendix A.

These results show that using linguistic features improves predictive performance as compared to when only learned features are used, and this improvement is consistent for all K and all data augmentation configurations. For order K of transitive extension, we note that moving beyond $K = 1$ yield minor improvements in performance. Without any

Table 3.10 Comparison of our model's performance with previously published performances on SemEval dataset

Model	Precision	Recall	F1-score
Das and Smith (2009) [13]	62.9	63.2	63.0
Guo and Diab (2012) [27]	58.3	52.5	65.5
Ji and Eisenstein (2013) [38]	66.4	62.8	64.5
Xu et al. (2014) [98]	72.2	72.6	72.4
Eyecioglu and Keller (2015) [20]	68.0	66.9	67.4
Zarella et al. (2015) [100]	56.9	**80.6**	66.7
Zhao and Lan (2015) [101]	**76.7**	58.3	66.2
Vo et al. (2015) [86]	68.5	63.4	65.9
Karan et al. (2015) [42]	64.5	67.4	65.9
Dey et al. (2016) [16]	75.6	72.6	74.1
Huang et al. (2017) [36]	64.3	65.7	65.0
Lan and Xu (2018) [50]	–	–	65.6
Agarwal et al. (2018) [1]	76.0	74.2	75.1
Our model	70.8	**80.6**	**75.4**

linguistic features, $K = 1$ produces the highest F1-score overall, with exception of just one configuration, i.e., P2, P3, P1, NP1, NP2. Similarly, maximum F1-score with linguistic features is also achieved with $K = 1$. These observations prove that in noisy text, full transitive closure can produce lower performances and the order K of transitive extension needs to be investigated to determine the optimal data augmentation strategy. Without linguistic features, as opposed to $K = *$, augmenting data with NP2 using $K = 1, 2$ or 3 does not drop the predictive performance of the model drastically but rather shows a noticeable improvement. However, with inclusion of linguistic features, NP2 augmentation has variable effect on the performance for each order of K.

We compare our results with existing work in terms of precision, recall and F1-score on test split in Table 3.10. On this dataset, [50] in their comparative study of re-implementing 7 different models, found that Pairwise Word Interaction Model (PWIM) [30] performs the best and achieves F1-score of 65.6%, which we report in the table. State of the art performance on the SemEval dataset is reported by [1] with F1-score of 75.1%. Our best performing model outperforms state of the art by achieving an F1-score of 75.4%.

3.4.4 Impact of Multiple Cascades

As all cascades are supervised, we can record predictive performances (on test set) for every cascade after each training epoch and compare them with that produced by discriminator

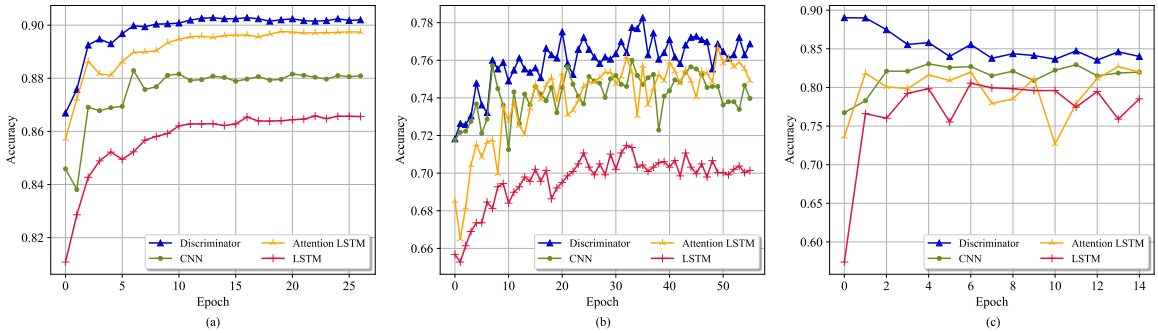


Fig. 3.5 Accuracy versus training epochs for each feature learning cascades and discriminator network on (a) Quora, (b) MSRP, and (c) SemEval datasets

network (complete multi-cascaded model). Figure 3.5 shows the performance of every cascade and the complete multi-cascaded model on Quora (plot (a)), MSRP (plot (b)), and SemEval (plot (c)) datasets.

It can be observed that the LSTM-based feature learner yields least performance and remains at the bottom throughout the training, while attention-based convolution feature learner is the next best performer. The attention-based LSTM learner is closely following the discriminator's performance but discriminator remains at the top during the training process after every epoch. This confirms that training a discriminator based on features learned from multiple perspectives is more fruitful as compared to relying on features learned by only one type of deep learning model. The same trend is followed in all the datasets. These results are presented only for the configuration which yields best results on each dataset.

3.4.5 Summary

Our extensive experiments evaluated our enhanced paraphrase detection model along several dimensions. These dimensions include noisy versus clean datasets, large versus small datasets, data augmentation steps and their variations, learned and linguistic features, and cascades in the multi-cascaded model. The key findings of our experiments are summarized below.

1. Data augmentation improves paraphrase detection predictive performance on all datasets (noisy, clean, large, and small). These easy steps can generate additional annotations that translate into the higher predictive performance from deep learning models.

2. Each step for generating additional paraphrase annotations produces an improvement in the prediction performance. On the other hand, only step NP1 for generating additional non-paraphrase annotations consistently improve performance; step NP2 can sometimes cause a decrease in predictive performance especially when the annotation is error-prone (e.g., the notion of paraphrase is not well defined, noisy text).

3. Linguistic features are important if a dataset is relatively small and noisy in nature. For such datasets, including linguistic features can produce significant boost in the predictive performance of our model.

4. When a dataset is sufficiently large, using linguistic features does not have any effect on the predictive performance.

5. For clean text, augmentation scheme of P2, P3, P1, and NP1 gives maximum performance in terms of F1-score on both datasets while for user-generated noisy text, this scheme yields maximum performance only with learned features only. When linguistic features are used, maximum performance is achieved without any augmentation.

6. It is not recommended to use full transitive closure ($K = *$) for user-generated noisy datasets as no noticeable and consistent improvement in performance is observed. For such datasets, data augmentation with the transitive extension should be investigated.

Chapter 4

Bilingual SMS Classification

Social media such as Facebook, Twitter, and Short Text Messaging Service (SMS) are popular channels for getting feedback from consumers on products and services. In Pakistan, with the emergence of e-government practices, SMS is being used for getting feedback from the citizens on different public services with the aim to reduce petty corruption and deficient delivery in services. Automatic classification of these SMS into predefined categories can greatly decrease the response time on complaints and consequently improve the public services rendered to the citizens. However, these SMS texts contain multilingual text written in the non-native script and informal diction (recall Chapter 1). Factors like informal verbiage, improper grammar, variation in spellings, code-switching, and short text length make the problem of automatic bilingual SMS classification highly challenging.

In this chapter, we present a modified version of multi-cascaded deep learning network *McM* for multi-class classification of bilingual short text (modifications are made in order to make McM compatible with single input as opposed to two inputs of paraphrase identification task). Our goal is to perform bilingual short text classification without any prior knowledge of the language, code-switching indication, language translation, normalizing lexical variations, or language transliteration. In multilingual text classification, previous approaches employ a single deep learning architecture, such as CNN or Long Short Term Memory (LSTM) for feature learning and classification. McM, on the other hand, employs three cascades (aka feature learners) to learn richer textual representations from three perspectives. These representations are then forwarded to a small discriminator network for final prediction. We compare the performance of the proposed model with existing CNN-based model for multilingual text classification [57]. We report a series of experiments using 3 kinds of embedding initialization approaches as well as the effect of attention mechanism [95].

The English language is well studied under the umbrella of NLP, hence many resources and datasets for the different problems are available. However, research on English-Roman

Urdu bilingual text lags behind because of non-availability of gold standard datasets. To this end, we present a large scale annotated dataset in Roman Urdu and English language with code-switching, for multi-class classification. The dataset consists of more than 0.3 million records and has been made available for future research.

The rest of the chapter is organized as follows. Section 4.1 introduces the dataset development process, preprocessing steps, and provides an explanation of the class labels assigned during the annotation process. In section 5.3, the modified architecture of the proposed model McM, its hyperparameters, evaluation metrics, and the experimental setup is discussed. Finally, we discuss the results of all variations of experiments and their comparison with baseline in section 4.3.

.

4.1 Dataset Acquisition and Description

The dataset consists of SMS feedbacks of the citizens of Pakistan on different public services availed by them. The objective of collecting these responses is to measure the performance of government departments rendering different public services. Preprocessing of the data is kept minimal. All records having only single word in SMS were removed as cleaning step. To construct the “gold standard", $313,813$ samples are manually annotated into 12 predefined categories by two annotators in supervision of a domain-expert. Involvement of the domain-expert was to ensure the practicality and quality of the “gold standard". Finally, stratified sampling method was opted for splitting the data into train and test partitions with $80-20$ ratio (i.e., 80% records for training and 20% records for testing). This way, training split has $251,050$ records while testing split has $62,763$ records. The rationale behind stratified sampling was to maintain the ratio of every class in both splits. The preprocessed and annotated data along with train and test split is made available [1]. Note that the department names and service availed by the citizens is mapped to an integer identifier for anonymity.

Class label ratios, corresponding labels, and it’s description are presented in Table 4.1.

4.2 Proposed Model and Experimentation

The proposed model *McM*, introduced in Chapter 3, is modified in order to take one input instead of two inputs. The linguistic features branch is also dropped. Rest of the model employ three feature learners (cascades) that are trained for classification independently (in parallel).

[1] https://github.com/haroonshakeel/bilingual_sms_classification

Table 4.1 Description of class label along with distribution of each class (in %) in the acquired dataset

Class label	Description	Class%
Appreciation	Citizen provided appreciative feedback.	43.1%
Satisfied	Citizen satisfied with the service.	31.1%
Peripheral complaint	Complains about peripheral service like non-availability of parking or complexity of the procedure.	8.2%
Demanded inquiry	More inquiry is required on the complaint.	5.7%
Corruption	Citizen reported bribery.	3.5%
Lagged response	Department responded with delay.	2.1%
Unresponsive	No response received by the citizen from the department.	2.0%
Medicine payment	Complainant was asked to buy basic medicine on his expense.	1.8%
Adverse behavior	Aggressive/intolerant behavior of the staff towards the citizen.	1.5%
Resource nonexistence	Department lacks necessary resources.	0.6%
Grievance ascribed	Malfeasance/Abuse of powers/official misconduct/sexual harassment to the complainant.	0.3%
Obnoxious/irrelevant	The SMS was irrelevant to public services.	0.2%

The input text is first mapped to embedding matrix of size $l \times d$ where l denotes the number of words in the text while d is dimensions of the embedding vector for each of these words. More formally, let $\mathcal{T} \in \{w_1, w_2, ..., w_l\}$ be the input text with l words, embedding matrix is defined by $X \in \mathbb{R}^{l \times d}$. This representation is then fed to three feature learners, which are trained with local supervision. The learned features are then forwarded to discriminator network for final prediction as shown in Fig. 4.1. Each of these components are discussed in subsequent subsections.

4.2.1 Stacked-CNN Learner

CNN learner is employed to learn n-gram features for identification of relationships between words. A 1-d convolution filter is used with a sliding window (kernel) of size k (number of n-grams) in order to extract the features. A filter W is defined as $W \in \mathbb{R}^{k \times d}$ for the

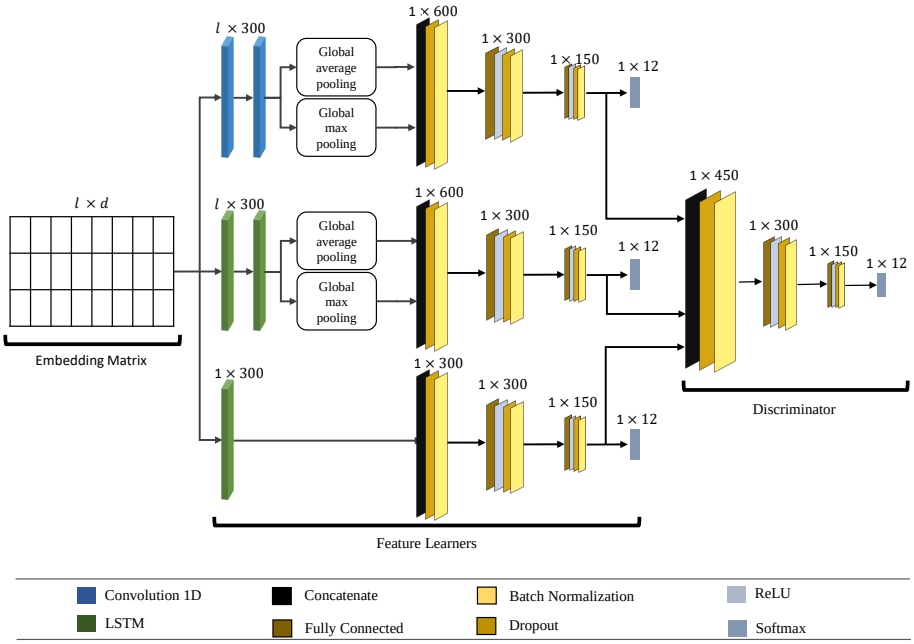


Fig. 4.1 Multi-cascaded model (McM) for bilingual short text classification (figure best seen in color)

convolution function. The word vectors starting from the position j to the position $j+k-1$ are processed by the filter W at a time. The window h_j is expressed as:

$$h_j = [X_j \oplus X_{j+1} \oplus, ..., \oplus X_{j+k-1}] \tag{4.1}$$

Where, the $\oplus$ represents the concatenation of word vectors. The number of filters are usually decided empirically. Each filter convolves with one window at a time to generate a feature map f_j for that specific window as:

$$f_j = \sigma(h_j \odot W + b) \tag{4.2}$$

Where, the $\odot$ represents convolution operation, b is a bias term, and σ is a nonlinear transformation function *ReLU*, which is defined as $\sigma(x) = max(x, 0)$. The feature maps of each window are concatenated across all filters to get a high level vector representation and fed as input to next CNN layer. Output of second CNN layer is followed by (i) global max-pooling to remove low activation information from feature maps of all filters, and (ii) global average-pooling to get average activation across all the n-grams.

These two outputs are then concatenated and forwarded to a small feedforward network having two fully-connected layers, followed by a *softmax* layer for prediction of this particular

learner. Dropout and batch-normalization layers are repeatedly used between both fully-connected layers to avoid features co-adaptation [77, 37].

4.2.2 Stacked-LSTM Learner

The traditional methods in deep learning do not account for previous information while processing current input. LSTM, however, is able to memorize past information and correlate it with current information [90]. LSTM structure has memory cells (aka LSTM cells) that store the information selectively. Each word is treated as one time step and is fed to LSTM in a sequential manner. While processing the input at current time step X_t, LSTM also takes into account the previous hidden state h_{t-1}. The LSTM represents each time step with an input, a memory, and an output gate, denoted as i_t, f_t and o_t respectively. The hidden state h_t of input X_t for each time step t is given by:

$$i_t = \sigma(W_i X_t + U_i h_{t-1} + b_i), \tag{4.3}$$

$$f_t = \sigma(W_f X_t + U_f h_{t-1} + b_f), \tag{4.4}$$

$$o_t = \sigma(W_o X_t + U_o h_{t-1} + b_o), \tag{4.5}$$

$$u_t = tanh(W_u + U_u h_{t-1} + b_u), \tag{4.6}$$

$$c_t = i_t * u_t + f_t * c_{t-1}, \tag{4.7}$$

$$h_t = o_t * tanh(c_t). \tag{4.8}$$

Where, the $*$ is element-wise multiplication and σ is sigmoid activation function.

Stacked-LSTM learner is comprised of two LSTM layers. Let H_1 be a matrix consisting of output vectors $\{h_1, h_2, ..., h_l\}$ that the first LSTM layer produced, denoting output at each time steps. This matrix is fed to second LSTM layer. Similarly, second layer produces another output matrix H_2 which is used to apply global max-pooling and global-average pooling. These two outputs are concatenated and forwarded to a two layered feedforward network for intermediate supervision (prediction), identical to previously described stacked-CNN learner.

4.2.3 LSTM Learner

LSTM learner is employed to learn long-term dependencies of the text as described in [90]. This learner encodes complete input text recursively. It takes one word vector at a time as input and outputs a single vector. The dimensions of the output vector are equal to the number of LSTM units deployed. This encoded text representation is then forwarded to

a small feedforward network, identical to aforementioned two learners, for intermediate supervision in order to learn features. This learner differs from stacked-LSTM learner as it learns sentence features, and not average and max features of all time steps (input words).

4.2.4 Discriminator Network

The objective of discriminator network is to aggregate features learned by each of above described three learners and squash them into a small network for final prediction. The discriminator employs two fully-connected layers with batch-normalization and dropout layer along with *ReLU* activation function for non-linearity. The *softmax* activation function with categorical cross-entropy loss is used on the final prediction layer to get probabilities of each class. The class label is assigned based on maximum probability. This is treated as final prediction of the proposed model. The complete architecture, along with dimensions of each output is shown in Fig. 4.1.

4.2.5 Experimental Setup

Pre-trained word embeddings on massive data, such as GloVe [67], give boost to predictive performance for multi-class classification [78]. However, such embeddings are limited to English language only with no equivalence for Roman Urdu. Therefore, in this study, we avoid using any *word-based* pre-trained embeddings to give equal treatment to words of each language. We perform three kinds of experiments. (1) Embedding matrix is constructed using ELMo embeddings [68], which utilizes characters to form word vectors and produces a word vector with $d = 1024$. We call this variation of the model McM_{E}. (2) Embedding matrix is initialized randomly for each word with word vector of size $d = 300$. We refer this particular model as McM_{R}. (3) We train domain specific embeddings[2] using word2vec with word vector of size $d = 300$ as suggested in original study [60]. We refer to this particular model as McM_{D}.

Furthermore, we also introduce soft-attention [95] between two layers of CNN and LSTM (in respective feature learner) to evaluate effect of attention on bilingual text classification. Attention mechanism “highlights" (assigns more weight) a particular word that contributes more towards correct classification. We refer to attention based experiments with subscript A for all three embedding initializations (i.e., McM_{EA}, McM_{RA}, McM_{DA}). This way, a total of 6 experiments (3 without attention and 3 with attention) are performed with different variations of the proposed model. To mitigate effect of random initialization of network

[2]These embeddings are also made available along with dataset.

weights, we fix the random seed across all experiments. We train each model for 20 epochs and create a checkpoint at epoch with best predictive performance on test split.

We re-implement the model proposed in [57], and use it as a baseline for our problem. The rationale behind choosing this particular model as a baseline is it's proven good predictive performance on multilingual text classification. For McM, the choices of number of convolutional filters, number of hidden units in first dense layer, number of hidden units in second dense layer, and recurrent units for LSTM are made empirically. Rest of the hyperparameters were selected by performing grid search using 20% stratified validation set from training set on McM_R. Available choices and final selected parameters are mentioned in Table 5.2. These choices remained same for all experiments and the validation set was merged back into training set.

Table 4.2 Hyperparameter tuning, the selection range, and final choice

Hyperparameter	Possible Values	Chosen Value
First CNN layer kernel size (k)	1, 2, 3, 4, 5	1
Second CNN layer kernel size (k)	1, 2, 3, 4, 5	2
Dropout rate	0.1, 0.2, 0.3, 0.4, 0.5	0.2
Optimizer	Adam, Adadelta, SGD	Adam
Learning rate	0.001, 0.002, 0.003, 0.004, 0.005	0.002

4.2.6 Evaluation Metrics

We employed the standard metrics that are widely adapted in the literature for measuring multi-class classification performance. These metrics are *accuracy, precision, recall,* and *F1-score*, where latter three can be computed using micro-average or macro-average strategies [76]. In micro-average strategy, each instance holds equal weight and outcomes are aggregated across all classes to compute a particular metric. This essentially means that the outcome would be influenced by the frequent class, if class distribution is skewed. In macro-average however, metrics for each class are calculated separately and then averaged, irrespective of their class label occurrence ratio. This gives each class equal weight instead of each instance, consequently favoring the under-represented classes.

In our particular dataset, it is more plausible to favor smaller classes (i.e., other than "Appreciation" and "Satisfied") to detect potential complaints. Therefore, we choose to report macro-average values for precision, recall, and F1-score which are defined by (4.9), (4.10),

and (4.11) respectively.

$$Precision = \frac{\sum_{i=1}^{C} \frac{TP_i}{TP_i+FP_i}}{C}, \tag{4.9}$$

$$Recall = \frac{\sum_{i=1}^{C} \frac{TP_i}{TP_i+FN_i}}{C}, \tag{4.10}$$

$$F1-score = \frac{\sum_{i=1}^{C} \frac{2\times Precision_i \times Recall_i}{Precision_i+Recall_i}}{C}. \tag{4.11}$$

4.3 Results and Discussion

Before evaluating the McM, we first tested the baseline model on our dataset. Table 5.3 presents results of baseline and all variations of our experiments. As each feature learner is supervised, we also show metrics for all three feature learners along with the final discriminator network. We focus our discussion on F1-score as accuracy is often misleading for dataset with unbalanced class distribution. However, for completeness sake, all measures are reported.

It is observed from the results that baseline model performs worst among all the experiments. The reason behind this degradation in performance can be traced back to the nature of the texts in the datasets (i.e., datasets used in original paper of baseline model [57] and in our study). The approach in base model measure the performance of the model on multilingual dataset in which there is no code-switching involved. The complete text belongs to either one language or the other. However, in our case, the SMS text can have code-switching between two language, variation of spelling, or non-standard grammar. Baseline model is simple 1 layered CNN model that is unable to tackle such challenges. On the other hand, McM learns the features from multiple perspectives, hence feature representations are richer, which consequently leads to a superior predictive performance. As every learner in McM is also supervised, all 4 components of the proposed model (i.e., stacked-CNN learner, stacked-LSTM learner, LSTM-learner, and discriminator) can also be compared with each other.

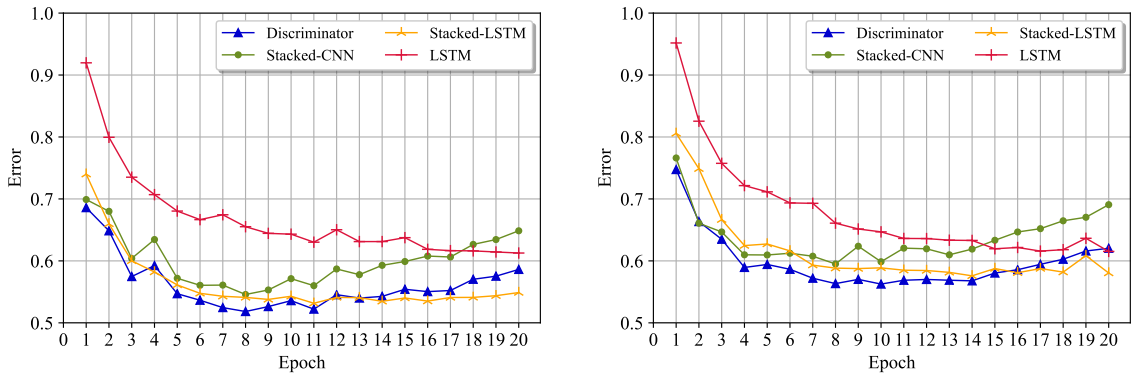
In our experiments, the best performing variation of the proposed model is McM_{D}. On this particular setting, discriminator is able to achieve an F1-score of 0.69 with precision and recall values of 0.72 and 0.68 respectively. Other components of McM also show the highest stats for all performance measures. However, for McM_{DA}, a significant reduction in performance is observed, although, attention-based models have been proven to show improvement in performance [95]. Investigating the reason behind this drop in performance is beyond the scope of this study. The model variations trained on ELMo embedding have

Table 4.3 Performance evaluation of variations of the proposed model and baseline. Showing highest scores in boldface.

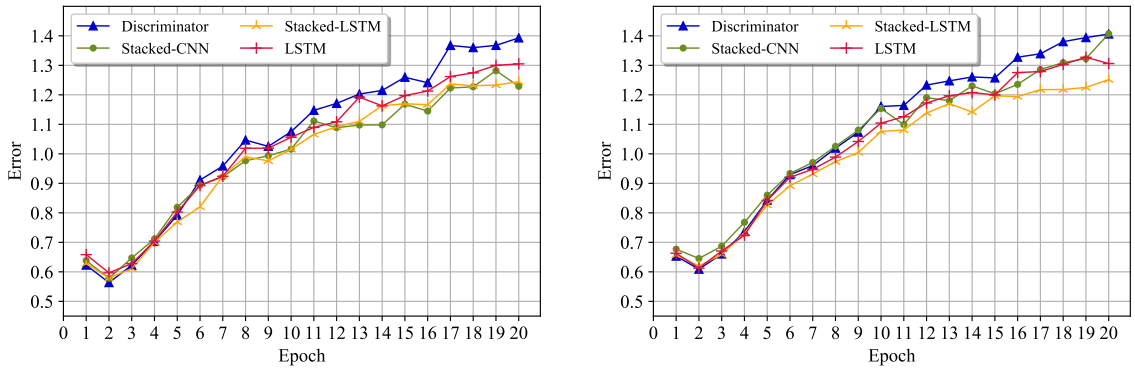
Model	Component	Accuracy	Precision	Recall	F1-score
Baseline [57]	-	0.68	0.52	0.37	0.39
McM_E	Stacked-CNN Learner	0.83	0.66	0.62	0.63
	Stacked-LSTM Learner	0.84	0.70	0.60	0.64
	LSTM Learner	0.80	0.69	0.48	0.51
	Discriminator	0.84	0.68	0.63	0.66
McM_{EA}	Stacked-CNN Learner	0.82	0.65	0.57	0.60
	Stacked-LSTM Learner	0.82	0.65	0.57	0.60
	LSTM Learner	0.80	0.62	0.49	0.51
	Discriminator	0.83	0.66	0.60	0.62
McM_R	Stacked-CNN Learner	0.82	0.66	0.59	0.62
	Stacked-LSTM Learner	0.82	0.66	0.58	0.61
	LSTM Learner	0.81	0.62	0.59	0.59
	Discriminator	0.83	0.64	0.61	0.62
McM_{RA}	Stacked-CNN Learner	0.80	0.65	0.52	0.53
	Stacked-LSTM Learner	0.81	0.65	0.55	0.58
	LSTM Learner	0.81	0.64	0.55	0.58
	Discriminator	0.81	0.64	0.58	0.59
McM_D	Stacked-CNN Learner	0.84	0.71	0.63	0.66
	Stacked-LSTM Learner	0.85	0.71	0.67	0.69
	LSTM Learner	0.83	0.68	0.60	0.63
	Discriminator	**0.86**	**0.72**	**0.68**	**0.69**
McM_{DA}	Stacked-CNN Learner	0.82	0.66	0.59	0.62
	Stacked-LSTM Learner	0.84	0.69	0.64	0.66
	LSTM Learner	0.83	0.67	0.61	0.63
	Discriminator	0.85	0.70	0.66	0.67

second highest performance. Discriminator of McM_E achieves an F1-score of 0.66, beating other learners in this experiment. However, reduction in performance is persistent when attention is used for McM_{EA}.

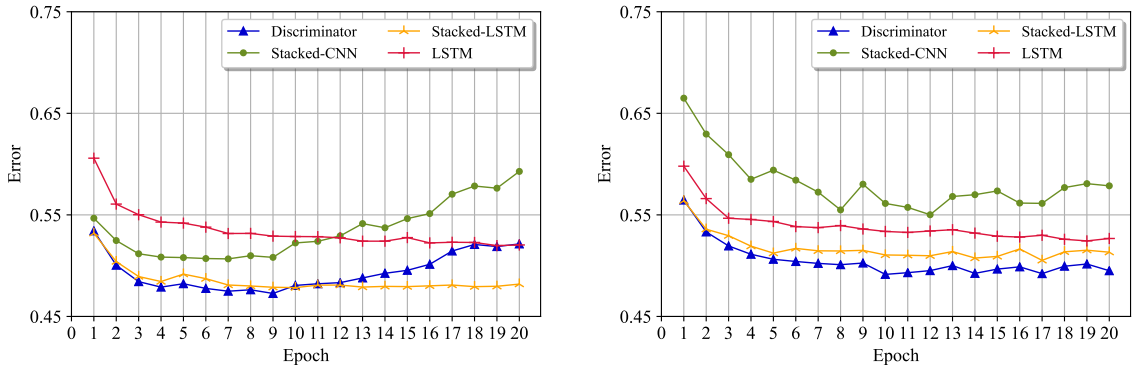
Regarding the experiments with random embedding initialization, McM_R shows similar performance to McM_{EA}, while McM_{RA} performs the worst. It is worth noting that in each experiment, discriminator network stays on top or performs equally as compared to other components in terms of F1-score. This is indication that discriminator network is able to



(a) McM_{E} (b) McM_{EA}



(c) McM_{R} (d) McM_{RA}



(e) McM_{D} (f) McM_{DA}

Fig. 4.2 Test error for all three feature learners and discriminator network over the epochs for all 4 variations of the model, showing lowest error for domain specific embeddings while highest for random embedding initialization.

learn richer representations of text as compared to methods where only single feature learner is deployed.

Furthermore, the results for testing error for each component (i.e., 3 learners and a discriminator network) for all 4 variations of the proposed model are presented in Fig. 4.2. It is evident that the least error across all components is achieved by McM_D model. Turning now to individual component performance, in ELMo embeddings based two models, lowest error is achieved by discriminator network, closely followed by stacked LSTM learner and stacked-CNN learner, while LSTM learner has the highest error. As far as model variations with random embeddings initializations are concerned, most interesting results are observed. As shown in subplot (c) and (d) in Fig. 4.2, McM_R and McM_{RA} tend to overfit. After second epoch, the error rate for all components of these two variations tend to increase drastically. However, it shows minimum error for discriminator in both variations, again proving that the features learned through multiple cascades are more robust and hold greater discriminative power. Note that in all 6 variations of experiments, the error of discriminator network is the lowest as compared to other components of McM. Hence it can be deduced that learning features through multiple perspectives and aggregating them for final prediction is more fruitful as compared to single method of learning.

Chapter 5

Sentiment Classification of Bilingual Short Text

Social media platforms are increasingly becoming popular for sharing sentiments towards a variety of topics. Thus, automatic sentiment classification of text posted on these platforms has become an important task to capture the overall “feel" of the masses towards a subject of interest. However, the posts on such platforms are often influenced by regional languages and are often noisy (recall Chapter 1). As a consequence, factors like informal diction, non-standard grammar, spelling variations, code-switching, acronyms, jargon, and short text length make the problem of automatic sentiment classification highly challenging.

It is established that pre-trained word embeddings, such as GloVe [67], give a boost to predictive performance of language models [78]. However, such *“word-based"* embeddings are limited to English language only with no equivalence for informal languages. An alternative, therefore, is to use *“character-based"* embedding [69, 82, 3]. Such embeddings are available in the form of pre-trained models on large scale data of English language, hence are well-suited for any language that uses English alphabets.

The focus of this chapter is sentiment classification *“Roman Urdu"* (RU) (refer to Chapter 1). Despite its prevalence, RU sentiment classification has got little attention and research on this *“language"* lags behind due to the non-availability of gold-standard datasets.

With respect to this particular task, our first contribution is that we develop an annotated dataset called *MultiSenti* for the problem of sentiment classification of RU short text.

Our second contribution is that we investigate the feasibility of adapting character-based pre-trained embedding models for sentiment classification of RU short text. To exhibit the contrast with adapted embeddings, we also train our own word-based multilingual embeddings on Roman Urdu corpus. The performance of both types of methods (i.e.,

adapting existing character-based embeddings vs training own word-based embeddings) in terms of sentiment classification accuracy is assessed exhaustively.

Our third contribution is that we propose two deep learning models for sentiment classification of RU short text, namely *McM* and *CNN-gram*. Recall from Chapter 4 that the former model employs three cascades (aka feature learners) to learn rich textual representations from three perspectives. These representations are then forwarded to a small discriminator network for final prediction. The latter model is inspired by [4] who showed that *n*-gram information are sufficient for multilingual short text sentiment classification. CNN-gram extracts *n*-gram information from the input text to detect sentiment. Both of these models tend to learn from raw text only without utilizing lexical normalization, language translation, language transliteration, or code-switching indication. The performance of the proposed models is compared with three existing multilingual sentiment classification models known as (i) ConvNet [57], (ii) Attention-LSTM [102], and (iii) SimpleConv [4]. These comparisons are made in terms of the F1-score on MultSenti dataset. The results demonstrate that McM outperforms other models in all of the experiments while CNN-gram achieves comparable results with existing models. The study also proves the practicality and usefulness of adapting character-based pre-trained embeddings from English language for Roman Urdu language.

The rest of the chapter is organized as follows. Section 5.1 describes MultiSenti dataset development process and its characteristics. Details on language resource adaptation are discussed in section 5.2. The proposed model architectures, their hyperparameters, implementation details, evaluation metrics, and domain-specific multilingual embeddings are introduced in section 5.3. We discuss the results of proposed models, their run time, and comparison with baseline models in section 5.4.

5.1 MultiSenti Dataset

The MultiSenti dataset is collected from Twitter during and after the general elections of Pakistan in the year 2018. The objective behind the collection of this dataset is to identify the overall emotion and sentiment of populous towards on-going election process and its result. The dataset has been categorized into *“negative"*, *“positive"*, and *“neutral"* sentiments. A sentiment in a tweet can either be expressed in monolingual or multilingual form. We focus on three types of manifestations for each sentiment namely, (i) *Roman Urdu*, (ii) *English*, and (iii) *Mixed*. *Romain Urdu* means that the sentiment is expressed in Romanized Urdu, *English* means that the sentiment is expressed in English, while *Mixed* means that the sentiment is expressed through both Roman Urdu and English text. Preprocessing of the data is kept minimal to the extent of lowering the cases and removing all the records

Table 5.1 Dataset characteristics

Class Label	Class %age	Language	Language %age
Negative	48.27%	Roman Urdu	46.34%
Positive	35.10%	English	2.52%
Neutral	16.63%	Mixed	51.14%

having only single word in tweet. The “gold standard" is constructed by manually annotating 20,735 samples into predefined categories by two annotators in supervision of a domain-expert. The involvement of the domain-expert was deemed necessary for the quality of the “gold standard". In case of conflict between two annotators, decision of domain expert was considered. Class labels percentages and language ratios in the dataset are presented in the Table 5.1. Finally, stratified sampling method was opted for splitting the data into train and test partitions with 80/20 ratio (i.e., 80% records for training and 20% records for testing). This way, training split has 16,588 records while testing split has 4,147 records. The rationale behind stratified sampling was to maintain the ratio of every class in both splits. The preprocessed and annotated data along with train and test split is made available publicly[1].

[1] https://github.com/haroonshakeel/multisenti

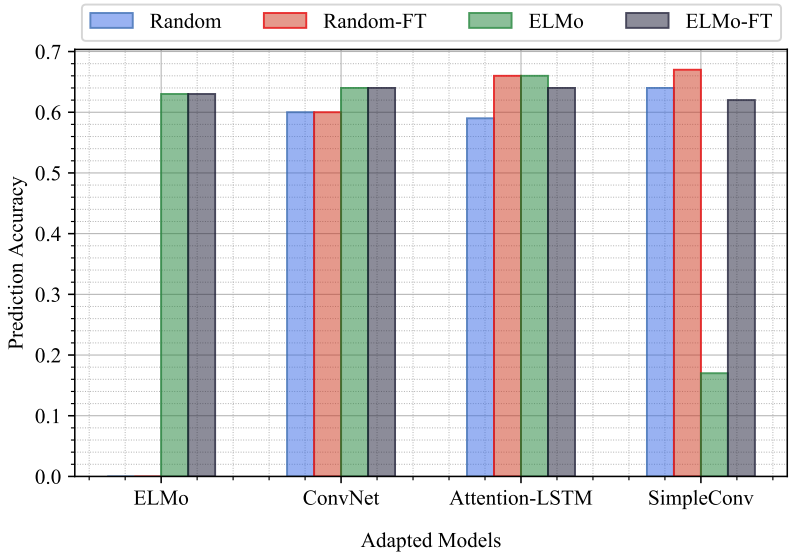


Fig. 5.1 Sentiment classification accuracy of each adapted model on MultiSenti dataset. FT = Finetuning. (figure best seen in color)

5.2 Language Resource Adaptation

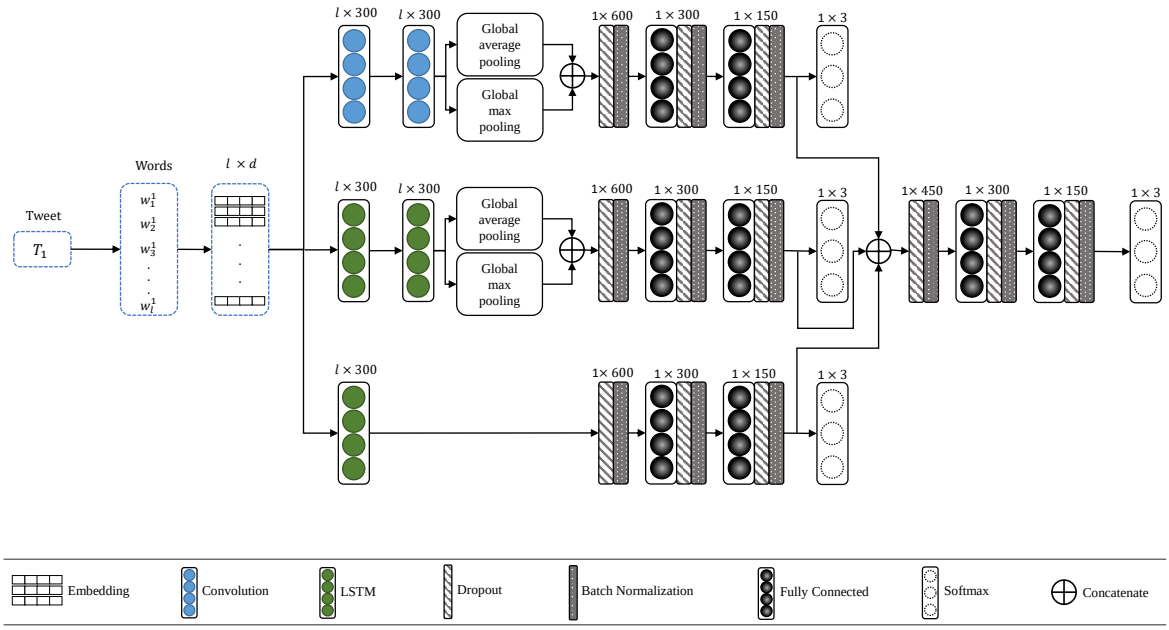


Fig. 5.2 Multi-cascaded model (McM) for sentiment classification of informal short text

We first opted to examine the feasibility of resource adaptation from other related studies. This includes deep learning models for multilingual sentiment classification and word embedding choices. As for the model choices, three models are selected from the literature that has proven good predictive performance on multilingual sentiment classification in under-resourced languages. These models include (i) ConvNets [57], (ii) Attention-LSTM [102], and (iii) SimpleConv [4]. All of these models were reimplemented using hyperparameters as defined in the original studies in order to validate their performance on the MultiSenti dataset.

As regards to word embedding choices for informal and unstructured short text, a well-known problem is the “out-of-vocabulary" where certain words are not found in the embedding base. In such cases, either random initialization of embeddings [4] or utilizing character-based pre-trained embeddings [3] is plausible. We investigate both strategies on MultiSenti dataset using all three deep learning models mentioned above. For our experiments, random embeddings are initialized from a uniform distribution with 300 dimensions. The choice of character-based pre-trained embeddings is restricted to ELMo [68], which is trained on a large-scale English language corpus and produces an embedding of 1024 dimensions. During the training of a model, embedding layer can be finetuned (i.e., weights of the embeddings are updated during training, aka “transfer-learning" of a pre-trained model)

or training can proceed without finetuning (i.e., where weights of the embeddings are not updated) [102]. These two cases are also analyzed to get more concrete insight into the adaptability. To assess the out-of-the-box performance of pre-trained embedding model, we also take prediction directly from ELMo by introducing a *softmax* layer on top of ELMo. This way, a total of 14 experiments were performed. The performance of each experiment was evaluated in terms of sentiment classification accuracy on test split of MultiSenti. Figure 5.1 compares the results of these tests.

The experiments reveal that ELMo out-of-the-box performs on par with the other variations, though finetuning does not affect its predictive performance. However, random embedding initialization benefits from finetuning on all three models. Slightly superior results are achieved when ConvNet and Attention-LSTM deep learning models are used on top of ELMo. Interestingly, SimpleConv model shows significant decline in the performance when ELMo embedding without finetuning was used. However, with finetuning, it was able to achieve comparable results with other variants. It is also worth noting that employing random embedding without finetuning yields lowest performance. These planned comparisons reveal that finetuning the embedding layer is more beneficial as compared to freezing the weight updates during the training, and using a deep model on top of an embedding layer is a perceptive choice.

However, all models on this informal language dataset underperform relative to the results reported on formal languages such as English, French, Greek, and Chinese [57, 102]. These observations clearly indicate that existing models for formal languages are not well-suited for informal language and call for novel model architectures specifically tailored for informal and unstructured language.

5.3 Proposed Solution

To learn the richer representations from the informal short text, two model architectures are proposed. Let $\mathcal{T} \in \{w_1, w_2, ..., w_l\}$ be the input text with l words, embedding matrix is defined by $X \in \mathbb{R}^{l \times d}$, where d is the dimensions of the embedding vector for each of the words. This matrix is then fed to the deep learning models. Details of both models are described in subsequent subsections.

5.3.1 Multi-cascaded Model (McM)

The proposed model, named multi-cascaded model *(McM)*, owes a lot to the findings by Reimers, N., & Gurevych (2017), who concluded that deeper models have negligible effect

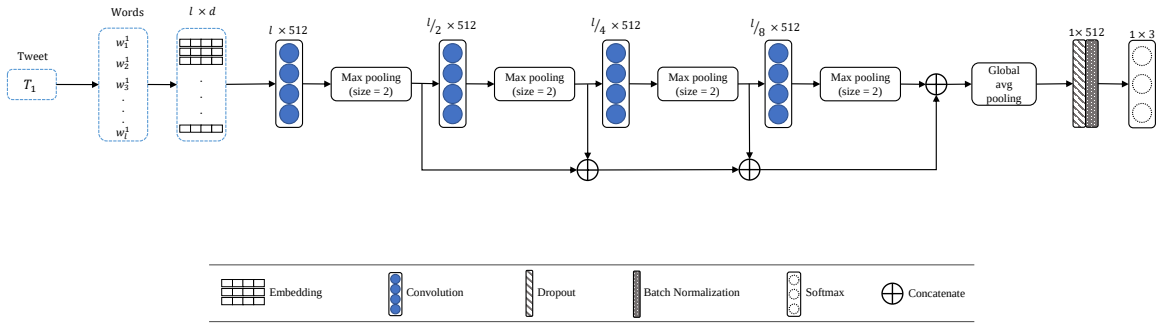


Fig. 5.3 CNN-gram model for sentiment classification of informal short text

on the predictive performance of NLP tasks [72]. McM exhibits a wider model, which employs three feature learners (cascades) that are trained for classification independently (in parallel) as shown in Figure 5.2. The learned features from these cascades are then forwarded to a discriminator network for final prediction. A brief description of each of these four components is given below.

Stacked-CNN Learner

CNN learner is employed to learn n-gram features for identification of relationships between words. A 1-d convolution filter is used with a sliding window (kernel) of size k (number of n-grams) in order to extract the features. A filter W is defined as $W \in \mathbb{R}^{k \times d}$ for the convolution function. The word vectors starting from the position j to the position $j+k-1$ are processed by the filter W at a time. The window h_j is expressed as:

$$h_j = [X_j \oplus X_{j+1} \oplus, ..., \oplus X_{j+k-1}] \tag{5.1}$$

Where the $\oplus$ represents the concatenation of word vectors. Each filter convolves with one window at a time to generate a feature map f_j for that specific window as:

$$f_j = \sigma(h_j \odot W + b) \tag{5.2}$$

Where $\odot$ represents convolution operation, b is a bias term and σ is a nonlinear transformation function $ReLU$, which is defined as $\sigma(x) = max(x, 0)$. The feature maps of each window are concatenated across all filters to get first level feature representation and fed as input to the next CNN layer. The output of second CNN layer is followed by (i) global max-pooling to remove low activation information from feature maps of all filters, and (ii) global average-pooling to get average activation across all the n-grams.

These two outputs are then concatenated and forwarded to a small feedforward network having two fully-connected layers, followed by a *softmax* layer for the prediction of this particular learner. Dropout and batch-normalization layers are repeatedly used between both fully-connected layers to avoid features co-adaptation [77, 37].

Table 5.2 Hyperparameters, the selection range, and final choice for each of the proposed model

Hyperparameter	Values Choices	Chosen for McM	Chosen for CNN-gram
First CNN layer kernel size (k)	1, 2, 3, 4, 5	1	-
Second CNN layer kernel size (k)	1, 2, 3, 4, 5	2	-
Dropout rate	0.1, 0.2, 0.3, 0.4, 0.5	0.5	0.4
Optimizer	RMSprop, Adam, Adadelta, SGD	Adam	Adadelta
Learning rate (Adam)	0.001, 0.002, 0.003, 0.004, 0.005	0.002	-
Learning rate (Adadelta)	0.5, 0.6, 0.7, 0.8, 0.9, 1.0	-	1.0

Stacked-LSTM Learner

The traditional methods in deep learning do not account for previous information while processing current input. LSTM, however, is able to memorize past information and correlate it with current information [90]. LSTM structure has memory cells (aka LSTM cells) that store the information selectively. Each word is treated as one time step and is fed to LSTM in a sequential manner. While processing the input at the current time step X_t, LSTM also takes into account the previous hidden state h_{t-1}. The LSTM represents each time step with an input, a memory, and an output gate, denoted as i_t, f_t and o_t respectively. The hidden state h_t of input X_t for each time step t is given by:

$$i_t = \sigma(W_i X_t + U_i h_{t-1} + b_i), \tag{5.3}$$

$$f_t = \sigma(W_f X_t + U_f h_{t-1} + b_f), \tag{5.4}$$

$$o_t = \sigma(W_o X_t + U_o h_{t-1} + b_o), \tag{5.5}$$

$$u_t = tanh(W_u + U_u h_{t-1} + b_u), \tag{5.6}$$

$$c_t = i_t * u_t + f_t * c_{t-1}, \tag{5.7}$$

$$h_t = o_t * tanh(c_t), \tag{5.8}$$

Where the $*$ is element-wise multiplication and σ is *sigmoid* activation function. Stacked-LSTM learner is comprised of two LSTM layers. The output of the first LSTM layer is fed to second LSTM layer and the output produced by second LSTM layer is forwarded to global

max-pooling and global-average pooling layers. The former drops the low activations while the latter averages activations across all time steps. These two outputs are concatenated and forwarded to a two-layered feedforward network for intermediate supervision (prediction), identical to previously described stacked-CNN learner.

LSTM Learner

LSTM learner is employed to learn long-term dependencies of the text as described in [90]. This learner encodes complete input text recursively. It takes one word vector at a time as input and returns a single vector. The dimensions of the output vector are equal to the number of LSTM units deployed. This encoded text representation is then forwarded to a small feedforward network identical to the aforementioned two learners, for intermediate supervision in order to learn features. This learner differs from stacked-LSTM learner as it learns sentence features, not average and max-features of all time steps (input words).

Discriminator Network

The objective of the discriminator network is to aggregate features learned by each of the above described three learners and squash them into a small network for final prediction. The discriminator employs two fully-connected layers with batch-normalization and dropout layer along with *ReLU* activation function for non-linearity. The *softmax* activation function with categorical cross-entropy loss is used on the final prediction layer to get probabilities of each class. The class label is assigned based on maximum probability. This is treated as the final prediction of the proposed model. The complete architecture, along with dimensions of each output is shown in Figure 5.2.

5.3.2 Convolutional Neural Network *n*-gram Model (CNN-gram)

In NLP, *n*-gram information can be used to learn a certain pattern from text [4]. The second proposed model named *CNN-gram* learns patterns based on unigram, bigram, trigram, and quadgram using CNN. Complete model architecture is illustrated in Figure 5.3. The first CNN layer uses kernel size of 1 with *ReLU* activation function to learn unigram representations followed by a max-pooling layer with pool size of 2. Max-pooling is utilized to drop low activation values from learned representations, which also acts as dimensionality reduction by downsampling the output. The max-pooled output is then forwarded to another CNN layer which uses kernel size of 2 to learn bigram patterns. This is followed by another max-pooling layer identical to first layer. Similarly, third and fourth CNN layers with max-pooling are used to learn trigram and quadgram patterns respectively. Note that these are not bigram,

trigram, and quadgram patterns in “true" sense as one of the two activations is dropped during max-pooling process with pool size of 2 after every CNN layer. However, on forwarded high activations, the notion of bigram, trigram, and quadgram holds true. Outputs of all four max-pooling layers are concatenated followed by a global average-pooling layer, which takes the average value as the feature corresponding to each filter. A dropout and batch-normalization layer is then utilized to avoid feature co-adaptation, followed by *softmax* activation function for final prediction of the sentiment. The categorical cross-entropy is used as loss function.

5.3.3 Hyperparameters Tuning

In both proposed models, the choices of the number of convolutional filters, number of units in dense layers, and number of LSTM units are made empirically. Figures 5.2 and 5.3 show these choices for McM and CNN-gram respectively. For both models, rest of the hyperparameters were selected by performing a grid search using a 20% stratified validation set taken from training set and utilizing random embedding initialization without finetuning. Available choices and final selected parameters are mentioned in Table 5.2. These choices remained same for all other experiments and the validation set was merged back into training set.

For McM, value for kernel size (k) for first and second CNN turned out to be 1 and 2 respectively. As for the dropout rate, 0.5 was optimal while Adam optimizer with a learning rate of 0.002 was selected. Turning now to CNN-gram, the kernel sizes for all CNN layers were fixed, as stated in section 5.3.2. While dropout rate, optimizer, and learning rate were chosen to be 0.4, Adadelta, and 1.0 respectively.

5.3.4 Multilingual Embeddings

Embeddings developed on domain-specific corpus can give a boost to the classification performance of the models [61, 78]. Keeping this in view, multilingual embeddings were also constructed from a combined corpus constituted of MultiSenti dataset and another large scale Roman Urdu dataset [2]. The total number of words in this combined corpus were more than 6.5 millions. We use skip-gram model of word2vec with word vector of size $d = 300$ as suggested in original study [60]. These embeddings are trained for $500,000$ iterations. By incorporating these embeddings for our experiments, we aim to answer the question *“Does it worth to train multilingual embeddings from informal language text when pre-trained embeddings on a large scale corpus of English language are readily available to adapt?"*

[2]These embeddings are made available along with dataset.

Table 5.3 Performance evaluation of variations of the proposed models and baselines. (Showing highest scores in boldface.)

		Without Finetuning				With Finetuning			
Model	**Embedding**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Accuracy**	**Precision**	**Recall**	**F1-score**
ELMo [68]	-	0.63	0.66	0.55	0.57	0.63	0.64	0.55	0.57
ConvNet [57]	Random	0.60	0.57	0.58	0.57	0.63	0.60	0.61	0.61
	ELMo	0.64	0.63	0.59	0.60	0.64	0.62	0.60	0.61
	Multilingual	0.63	0.60	0.57	0.58	0.65	0.62	0.60	0.61
Attention-LSTM [102]	Random	0.59	0.56	0.56	0.55	0.66	0.64	0.64	0.64
	ELMo	0.66	0.64	0.61	0.62	0.64	0.63	0.60	0.61
	Multilingual	0.64	0.61	0.61	0.61	0.66	0.63	0.62	0.62
SimpleConv [4]	Random	0.64	0.63	0.58	0.60	0.67	0.65	0.63	0.63
	ELMo	0.17	0.06	0.33	0.10	0.62	0.65	0.54	0.55
	Multilingual	0.35	0.12	0.33	0.17	0.35	0.12	0.33	0.17
McM	Random	0.67	**0.67**	0.61	0.62	0.67	0.69	0.60	0.62
	ELMo	0.67	0.66	0.62	0.63	0.68	0.65	**0.65**	**0.65**
	Multilingual	**0.68**	**0.67**	**0.64**	**0.65**	**0.69**	**0.71**	0.62	0.64
CNN-gram	Random	0.49	0.32	0.38	0.35	0.64	0.68	0.57	0.59
	ELMo	0.66	**0.67**	0.61	0.63	0.65	0.66	0.59	0.60
	Multilingual	0.65	0.63	0.59	0.60	0.66	0.66	0.60	0.62

5.3.5 Evaluation Metrics

We employed the standard metrics that are widely adopted in the literature for measuring sentiment classification performance involving more than two classes. These metrics are *accuracy, precision, recall,* and *F1-score*, where latter three can be computed using micro-average or macro-average strategies [76, 4]. In the micro-average strategy, the outcome for any metric is influenced by the majority class, if the distribution is skewed. Therefore, it is plausible to use macro-average strategy which is insensitive to skewness in class distribution. In MultiSenti dataset, “neutral" class is underrepresented, hence we choose to report macro-average values for precision, recall, and F1-score which are defined by (5.9), (5.10), and (5.11) respectively.

$$Precision = \frac{\sum_{i=1}^{C} \frac{TP_i}{TP_i+FP_i}}{C}, \tag{5.9}$$

$$Recall = \frac{\sum_{i=1}^{C} \frac{TP_i}{TP_i+FN_i}}{C}, \tag{5.10}$$

$$F1-score = \frac{\sum_{i=1}^{C} \frac{2\times Precision_i \times Recall_i}{Precision_i+Recall_i}}{C}. \tag{5.11}$$

Where TP, FP, and FN stand for *true positives, false positives* and *false negatives* respectively, while C is the number of unique classes in the dataset.

5.3.6 Implementation Details

All the implementation is done in Python using Keras library with Tensorflow backend. All weights of the networks are initialized randomly and to mitigate the effect of randomness, random seed is fixed across all experiments. In each of the experiments, the model is trained for 100 epochs. A checkpoint of the learned weights is saved at epoch with best predictive performance on the test split. The early stopping approach is also opted and training is stopped if testing error does not decrease for 10 epochs.

5.4 Results and Discussion

This section summarizes the main findings of the work. The results of both of the proposed models are compared with existing multilingual sentiment classification models described in section 5.2. We focus the discussion on F1-score of all the models. However, a comprehensive comparison of all the metrics is given in Table 5.3. For each variation of the embedding, results for both cases (i.e.,with and without embedding finetuning) are presented. Based on the results, we make following observations.

Using pre-trained embeddings out-of-the-box yields identical performance when used either without or with finetuning. Specifically talking about the case when a model is used on top of the embeddings, ELMo embeddings without finetuning outperform the random and multilingual embeddings on ConvNet, Attention-LSTM, and CNN-gram models. Interestingly, in the case of SimpleConv model, ELMo yields poorest performance. Further examination of this particular case revealed that model was unable to learn when any pre-trained embeddings were used. However, using random embeddings for this particular model gives output comparable to other models. This implies that the model was specifically engineered to work with random embedding (as is evident from the original study). Regarding the use of random embedding for other models, the proposed model McM achieves highest F1-score, while the proposed CNN-gram model yields the poorest performance. Amongst rest of the models, ConvNet marginally outperforms Attention-LSTM. Regarding the use of multilingual embeddings, it was found that the least F1-score was achieved by SimpleConv, while McM achieved the highest score, which surpasses all the experiments without finetuning the embeddings.

Turning now to the case of finetuning, ConvNet performs identical in terms of F1-score for all of the embeddings, while Attention-LSTM and SimpleConv benefit from finetuning when random embeddings are used. For the proposed model CNN-gram, multilingual embeddings yield the highest score. If we now turn to McM model, ELMo embeddings yield the highest F1-score of 0.65. This is an interesting finding as it is identical to the F1-score

Table 5.4 Language wise breakdown of predictive performance of each model (in terms of % correctly classified)

		Without Finetuning			With Finetuning		
Model	**Embedding**	**Roman Urdu**	**English**	**Mixed**	**Roman Urdu**	**English**	**Mixed**
ELMo [68]	-	63.57%	67.88%	63.09%	62.54%	65.14%	62.71%
ConvNet [57]	Random	58.74%	58.72%	60.35%	63.63%	65.14%	62.29%
	ELMo	64.41%	64.22%	63.85%	63.99%	66.05%	63.42%
	Multilingual	64.72%	58.71%	61.44%	66.44%	59.63%	63.28%
Attention-LSTM [102]	Random	59.78%	54.13%	58.88%	67.01%	67.89%	65.78%
	ELMo	66.13%	69.72%	64.65%	65.30%	64.22%	63.52%
	Multilingual	65.71%	62.38%	62.90%	67.22%	63.30%	64.32%
SimpleConv [4]	Random	64.41%	60.55%	64.08%	68.10%	66.97%	65.87%
	ELMo	16.54%	33.94%	15.83%	62.80%	62.38%	62.09%
	Multilingual	31.63%	39.44%	37.99%	31.63%	39.44%	37.99%
McM	Random	68.16%	66.97%	66.20%	68.26%	69.72%	66.54%
	ELMo	66.91%	68.80%	67.43%	68.78%	**72.47%**	67.06%
	Multilingual	**69.04%**	66.05%	**67.81%**	**70.49%**	61.46%	**68.01%**
CNN-gram	Random	58.49%	39.44%	50.70%	66.44%	61.46%	62.61%
	ELMo	66.18%	**70.64%**	66.30%	66.33%	68.80%	63.75%
	Multilingual	67.48%	57.79%	62.66%	67.48%	57.79%	65.64%

of McM when multilingual embedding was used. As CNN-gram model ignores longterm dependencies and only accounts for *n*-gram information, it performs worst than McM but is comparable with other models.

Figure 5.4 presents average time taken per epoch (at log scale; because values for simple models like ConvNets were so small, that these were difficult to capture at regular scale). With respect to multilingual embedding, the pre-training time of $500,000$ iterations is not part of this analysis. It is worth noting that even though simpler networks such as ConvNets and SimpleConv take the least amount of time, their performance is inconsistent across all settings. While the proposed model McM shows the highest performance in majority of the cases with $\pm 3\%$ variation for each embedding. These findings lead to conclude that no apparent advantage exists in training word-based multilingual embeddings from scratch. The pre-trained character-based embedding on the English language with finetuning suffices for informal language to get identical results while avoiding pre-training overhead. However, to get most out of these embedding, a carefully tailored model for sentiment classification of informal short text is crucial.

As was mentioned in section 5.1, a particular tweet can be entirely in Roman Urdu, English, or both languages. Table 5.4, presents a more detailed analysis of the experiments and compares the performance of the models in terms of language-wise prediction accuracy, that is; how many instances of a specific language were correctly classified. It is worthwhile

to note that all the models were trained on all the languages combined. The language-wise analysis sheds light on the model's tendency to favor a particular language. It is found that, in general, ELMo embedding tends to favor the English language, which sustains the intuitive sense as these embeddings are trained on English language. It is interesting to note that multilingual embedding tends to favor Roman Urdu (and consequently Mixed) language. However, this does not affect the model's performance drastically, hence it does not help to draw a strong conclusion that training a domain-specific embedding is always favorable. Furthermore, training the embedding from scratch is a costly approach and as results suggest, the performance gains for the models are not worthwhile. One can argue that embeddings trained on an informal multilingual corpus, which is comparable in size to the corpus of English language, could yield better performance than adapting the embeddings. However, this leads to the initial paradox of not having enough data resources for the informal languages.

Table 5.5 lists a few examples from the test split of the MultiSenti dataset and their corresponding predicted label from the best-performing variation of each model. Translation of each tweet in English is given in square braces to enhance the readability. It is observed that for all selected tweets, irrespective of language, ELMo fails to correctly classify them. Sarcastic tweets in Roman Urdu are misclassified by all of the models, which demands to investigate the phenomenon further, this, however, is beyond the scope of this study. In general, the proposed models perform better for shorter tweets. However, longer tweets in

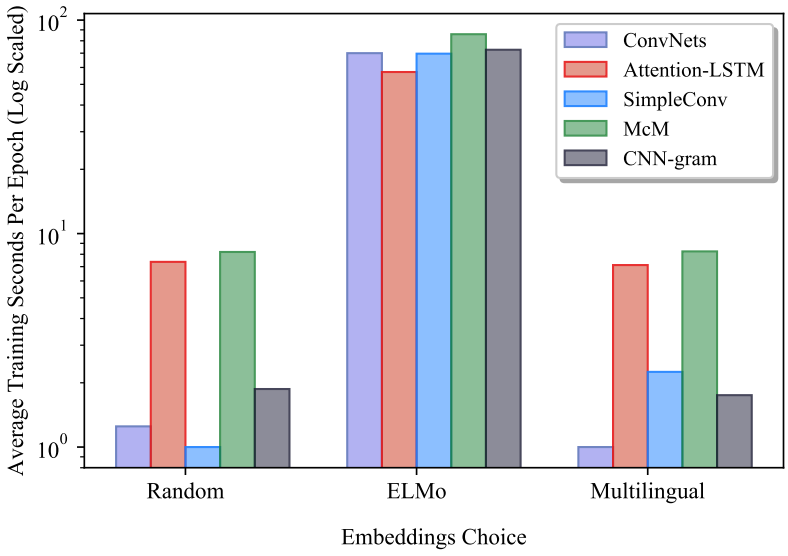


Fig. 5.4 Average training time per epoch (in seconds) for each configuration. (figure best seen in color)

Table 5.5 Predictions of best performing variation of each model on specific tweets. (RU : Roman Urdu, E : English, M : Mixed, - : negative, + : positive, 0 : neutral)

Tweet [Translation]	**Lang.**	**Ground Truth**	**ELMo**	**Conv-Net**	**Attention-LSTM**	**Simple-Conv**	**McM**	**CNN-gram**
ik ki bongian [ik and his foolishness]	RU	-	+	+	0	0	+	0
aur karo burayi [malign more]	RU	-	+	+	-	0	-	-
pmln mera janon meri jan [pmln my passion my life]	RU	+	-	+	+	+	+	+
sialkot sy b pmln jeet gya tottaly [pmln has totally won from sialkot also]	M	0	-	0	0	0	0	0
i agree with this bhai log vote daal rahay hotay to itni seats jeetna mushkil hota [i agree with this it would've been difficult to win these much seats if gangsters were voting]	M	+	-	-	-	-	-	-
ik today in banigala	E	0	+	0	0	0	0	0
has aleem khan won punjab seat	E	0	+	0	+	0	0	+

any language that are nuanced (like using "if" and "but" to negate what was said before) are not classified correctly by any classifier. These insights open up avenues for future research in this direction.

Chapter 6

Conclusion and Future Work

In this thesis, we present a novel deep learning model architecture called multi-cascaded model (McM). The model employs multiple feature learners to encode and classify short texts. As such, it exploits multiple semantic cues to distinguish between the classes. The proposed multi-cascaded model is both deep and wide in architecture, and it embodies previous best practices in deep models. We show its utility for three short text analytics tasks.

First, the effectiveness of the proposed model is shown on the task of paraphrase identification. We carry out an ablation study where one cascade is ignored at a time to justify the utilization of multiple cascades. The results confirm that multiple learners are able to produce a more robust model as compared to the single learner. We evaluate and compare our model on three benchmark datasets of short texts in the English language, representing both noisy and clean text. Our model produces a higher predictive performance on all three datasets beating all previously published results on them. The proposed model exhibits generalization across both noisy and clean texts. As deep learning models are inherently data-hungry, we also present a data augmentation strategy to enhance the model's performance. The data augmentation strategy generates additional paraphrase and non-paraphrase annotations based on the graph analysis of the existing annotations. We find the data augmentation strategy useful to enhance the model's predictive performance. We also study the impact of different steps in data augmentation and the use of linguistic features in conjunction with learned features. We found that linguistic features can improve the performance of the model when the dataset has too few records (a couple of thousands) while they are ineffectual when the dataset is sufficiently large (having a couple of hundred thousand records). With regards to the data augmentation strategy, we find that it yields significant improvement in the performance of the proposed model for paraphrase detection on a clean text dataset. However, noisy text demands caution while employing the proposed data augmentation strategy. To this end, we parametrize the strategy to control the level of augmentation. It

is plausible that the augmentation strategy inflicts conflicts in annotation, where a pair of text is declared as a paraphrase, while in actual annotations, it is non-paraphrase and vice versa. The subjective analysis of such conflicts highlighted that the annotation produced by data augmentation is correct, while the existing annotation from crowd-source was erroneous. In the future, it would be beneficial to carry out a more in-depth analysis of conflicts and investigate strategies for resolving them. Furthermore, this strategy can also be adapted in intelligent crowd-sourcing platforms, where pairs of texts are labeled in order to improve the annotation quality and detect errors in real-time.

Second, we show the efficacy of McM for multi-class classification of bilingual (English-Roman Urdu) text with code-switching. For this purpose, a new dataset of $313,813$ from SMS feedbacks is developed. The dataset is intended for the enhancement of petty corruption detection in public offices and to provide grounds for future research in this direction. Furthermore, three word embedding initialization techniques and the soft-attention mechanism is also investigated. The observations from extensive experimentation led us to conclude that: (1) word embeddings vectors generated through characters tend to favor bilingual text classification as compared to random embedding initialization, (2) the attention mechanism tends to decrease the predictive performance of the model, irrespective of embedding types used, (3) using features learned through single perspective yield poor performance for bilingual text with code-switching, (4) training domain-specific embeddings on a large corpus and using them to train the model achieves the highest performance. With regards to future work, it is worthwhile to investigate the reason behind the degradation of model performance with soft-attention, as in literature, attention is known to improve the performance of the model.

Third, the advantage of using McM for the task of sentiment classification in the code-switched informal short text (tweets in Roman Urdu) is demonstrated. For this purpose, we develop another dataset of $20,735$ tweets that are collected during the general elections of Pakistan in the year 2018. For this task, we investigate the feasibility of adapting embeddings and language models from resource-rich languages. Furthermore, we also propose another model inspired by findings in [4] and compare it with McM and three other baselines. Our work has led us to conclude that adapting existing resources from a resource-rich language to an informal language is practical. The evidence from this study suggests that this is only true for the tasks that demand the availability of large scale corpus, such as training embeddings. It is evident from the results that an embedding trained on sufficiently large corpus in the English language can successfully be adapted for an informal language. However, this is not necessarily true for the model choice. It is crucial that a model is explicitly engineered towards a colloquial language as compared to adapting models developed for other languages. Our work has a limitation that it only investigates the adaptation of one character-based

embedding, while there exist different character-based pre-trained embeddings trained on the English language, such as BERT [15] (for which a multilingual variant is also made available recently). To further this research, we plan to investigate other embedding choices. Given the results of the current extensive experimentation, we hope that further tests will be in line with the findings of this study.

In summary, in this research, we have highlighted the importance of text analytics for freestyle, informal, and multilingual short text. In particular, we have presented a novel deep learning architecture that learns the context and semantics of user-generated clean and noisy short text from multiple perspectives. In our presentation, we demonstrate the effectiveness and efficiency of the proposed model on three tasks, i.e., (1) paraphrase identification, (2) multilingual multi-class text classification, and (3) sentiment classification of code-switched short text. For the last two tasks, two large-scale annotated datasets are developed and made publicly available to further the research in this direction. We also compare different embedding initialization approaches, and our findings clarify that a carefully tailored model plays a more critical role than embedding initialization for a robust text analytics system. We have devised a systematic data augmentation strategy for paraphrase identification task. Our results suggest that employing this strategy reliably generates additional paraphrase and non-paraphrase annotations from existing annotations, consequently improving the performance of the model. Furthermore, we also show the utility of the proposed augmentation strategy to identify erroneous annotation in the dataset.

This research has taken a step closer to language-independent text analytics, which we are currently in the process of investigating.

References

[1] Agarwal, B., Ramampiaro, H., Langseth, H., and Ruocco, M. (2018). A deep network model for paraphrase detection in short text messages. *Information Processing & Management, (IPM)*, 54(6):922–937.

[2] Argamon, S., Koppel, M., Pennebaker, J. W., and Schler, J. (2009). Automatically profiling the author of an anonymous text. *Communications of the ACM*, 52(2):119–123.

[3] Arora, M. and Kansal, V. (2019). Character level embedding with deep convolutional neural network for text normalization of unstructured data for twitter sentiment analysis. *Social Network Analysis and Mining, (SNAM)*, 9(1):1–14.

[4] Attia, M., Samih, Y., Elkahky, A., and Kallmeyer, L. (2018). Multilingual multi-class sentiment classification using convolutional neural networks. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 635–640.

[5] Barrón-Cedeño, A., Vila, M., Martí, M. A., and Rosso, P. (2013). Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection. *Computational Linguistics, (CL)*, 39(4):917–947.

[6] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research, (JMLR)*, 3:1137–1155.

[7] Bogdanova, D., dos Santos, C., Barbosa, L., and Zadrozny, B. (2015). Detecting semantically equivalent questions in online user forums. In *Proceedings of the Conference on Computational Natural Language Learning, (CNNL)*, pages 123–131.

[8] Cagnina, L., Errecalde, M., Ingaramo, D., and Rosso, P. (2014). An efficient particle swarm optimization approach to cluster short texts. *Information Sciences*, 265:36–49.

[9] Chen, C.-H., Lee, W.-P., and Huang, J.-Y. (2018a). Tracking and recognizing emotions in short text messages from online chatting services. *Information Processing & Management (IPM)*, 54(6):1325–1344.

[10] Chen, X., Sun, Y., Athiwaratkun, B., Cardie, C., and Weinberger, K. (2018b). Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics, (TACL)*, 6:557–570.

[11] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

[12] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[13] Das, D. and Smith, N. A. (2009). Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the Annual Meeting of the ACL and the International Joint Conference on Natural Language Processing of the AFNLP, (ACL-IJCNLP)*, volume 1, pages 468–476.

[14] Denecke, K. (2008). Using sentiwordnet for multilingual sentiment analysis. In *International Conference on Data Engineering Workshop*, pages 507–512.

[15] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics (NACACL): Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

[16] Dey, K., Shrivastava, R., and Kaushik, S. (2016). A paraphrase and semantic similarity detection system for user generated short-text content on microblogs. In *Proceedings of the International Conference on Computational Linguistics, (COLING)*, pages 2880–2890.

[17] Dolan, B., Quirk, C., and Brockett, C. (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the International Conference on Computational Linguistics, (COLING)*, pages 350–357.

[18] dos Santos, C., Barbosa, L., Bogdanova, D., and Zadrozny, B. (2015). Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, (ACL-IJCNLP)*, pages 694–699.

[19] El-Alfy, E.-S. M., Abdel-Aal, R. E., Al-Khatib, W. G., and Alvi, F. (2015). Boosting paraphrase detection through textual similarity metrics with abductive networks. *Applied Soft Computing, (ASC)*, 26:444–453.

[20] Eyecioglu, A. and Keller, B. (2015). Twitter paraphrase identification with simple overlap features and svms. In *Proceedings of the International Workshop on Semantic Evaluation, (SemEval)*, pages 64–69.

[21] Fader, A., Zettlemoyer, L., and Etzioni, O. (2013). Paraphrase-driven learning for open question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, (ACL) (Long Papers)*, volume 1, pages 1608–1618.

[22] Fatima, M., Anwar, S., Naveed, A., Arshad, W., Nawab, R. M. A., Iqbal, M., and Masood, A. (2018). Multilingual sms-based author profiling: Data and methods. *Natural Language Engineering, (NLE)*, 24(5):695–724.

[23] Ferreira, R., Cavalcanti, G. D., Freitas, F., Lins, R. D., Simske, S. J., and Riss, M. (2018). Combining sentence similarities measures to identify paraphrases. *Computer Speech & Language, (CSL)*, 47:59–73.

[24] Figueroa, A. and Neumann, G. (2013). Learning to rank effective paraphrases from query logs for community question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence, (AAAI)*, pages 1099–1105.

[25] Gao, W., Peng, M., Wang, H., Zhang, Y., Xie, Q., and Tian, G. (2019). Incorporating word embeddings into topic modeling of short text. *Knowledge and Information Systems, (KIS)*, 61(2):1123–1145.

[26] Gillick, D., Brunk, C., Vinyals, O., and Subramanya, A. (2016). Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306.

[27] Guo, W. and Diab, M. (2012). Modeling sentences in the latent space. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, (ACL): Long Papers*, volume 1, pages 864–872.

[28] Han, B. and Baldwin, T. (2011). Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 368–378.

[29] He, H., Gimpel, K., and Lin, J. (2015). Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, (EMNLP)*, pages 1576–1586.

[30] He, H. and Lin, J. (2016). Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, (NAACL): Human Language Technologies*, pages 937–948.

[31] Heikinheimo, H. and Ukkonen, A. (2013). The crowd-median algorithm. In *Association for the Advancement of Artificial Intelligence Conference on Human Computation and Crowdsourcing, (HCOMP)*, pages 69–77.

[32] Heilman, M. and Smith, N. A. (2010). Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, (NAACL): Human Language Technologies*, pages 1011–1019.

[33] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[34] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9:1735–1780.

[35] Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In *Proceedings of the International Conference on Neural Information Processing Systems, (NIPS)*, pages 2042–2050.

[36] Huang, J., Yao, S., Lyu, C., and Ji, D. (2017). Multi-granularity neural sentence model for measuring short text similarity. In *International Conference on Database Systems for Advanced Applications, (DASFAA)*, pages 439–455.

[37] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456.

[38] Ji, Y. and Eisenstein, J. (2013). Discriminative improvements to distributional sentence similarity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, (EMNLP)*, pages 891–896.

[39] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning, (ECML)*, pages 137–142.

[40] Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

[41] Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665.

[42] Karan, M., Glavaš, G., Šnajder, J., Bašić, B. D., Vulić, I., and Moens, M.-F. (2015). Tkl-bliir: Detecting twitter paraphrases with tweetingjay. In *Proceedings of the International Workshop on Semantic Evaluation, (SemEval)*, pages 70–74.

[43] Kenter, T. and De Rijke, M. (2015). Short text similarity with word embeddings. In *Proceedings of the International Conference on Information and Knowledge Management, (CIKM)*, pages 1411–1420.

[44] Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *13th AAAI Conference on Artificial Intelligence*.

[45] Kleindessner, M. and von Luxburg, U. (2017). Kernel functions based on triplet comparisons. In *Proceedings of the International Conference on Neural Information Processing Systems, (NIPS)*, pages 6807–6817.

[46] Kleindessner, M. and Von Luxburg, U. (2017). Lens depth function and k-relative neighborhood graph: Versatile tool for ordinal data analysis. *Journal of Machine Learning Research, (JMLR)*, 18(1):1889–1940.

[47] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., and Brown, D. (2019). Text classification algorithms: A survey. *Information*, 10(4):150.

[48] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems, (NIPS)*, pages 1097–1105.

[49] Kusner, M. J., Sun, Y., Kolkin, N. I., and Weinberger, K. Q. (2015). From word embeddings to document distances. In *Proceedings of the International Conference on Machine Learning, (ICML)*, pages 957–966.

[50] Lan, W. and Xu, W. (2018). Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of the International Conference on Computational Linguistics, (COLING)*, pages 3890–3902.

[51] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning, (ICML)*, pages 1188–1196.

[52] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research, (JMLR)*, 5:361–397.

[53] Liu, F., Weng, F., and Jiang, X. (2012). A broad-coverage normalization system for social media language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1035–1044.

[54] Liu, X., Zhang, S., Wei, F., and Zhou, M. (2011). Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 359–367.

[55] Madnani, N., Tetreault, J., and Chodorow, M. (2012). Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, (NAACL): Human Language Technologies*, pages 182–190.

[56] Maharjan, S., Blair, E., Bethard, S., and Solorio, T. (2015). Developing language-tagged corpora for code-switching tweets. In *Proceedings of The Linguistic Annotation Workshop, (LAW)*, pages 72–84.

[57] Medrouk, L. and Pappa, A. (2017). Deep learning model for sentiment analysis in multi-lingual corpus. In *International Conference on Neural Information Processing (ICONIP)*, pages 205–212.

[58] Medrouk, L. and Pappa, A. (2018). Do deep networks really need complex modules for multilingual sentiment polarity detection and domain classification? In *International Joint Conference on Neural Networks, (IJCNN)*, pages 1–6.

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

[60] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Neural Information Processing Systems, (NIPS)*, pages 3111–3119.

[61] Mogadala, A. and Rettinger, A. (2016). Bilingual word embeddings from parallel and non-parallel corpora for cross-language text classification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, (NACACL): Human Language Technologies*, pages 692–702.

[62] Mohammad, A.-S., Jaradat, Z., Mahmoud, A.-A., and Jararweh, Y. (2017). Paraphrase identification and semantic text similarity analysis in arabic news tweets using lexical, syntactic, and semantic features. *Information Processing & Management, (IPM)*, 53(3):640–652.

[63] Nie, Y. and Bansal, M. (2017). Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the Workshop on Evaluating Vector Space Representations for NLP, (RepEval@EMNLP)*, pages 41–45.

[64] Oliva, J., Serrano, J. I., del Castillo, M. D., and Iglesias, A. (2011). Symss: A syntax-based measure for short-text semantic similarity. *Data and Knowledge Engineering, (DKE)*, 70(4):390–405.

[65] Pagliardini, M., Gupta, P., and Jaggi, M. (2018). Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, (NAACL): Human Language Technologies*, pages 528–540.

[66] Paltoglou, G. and Thelwall, M. (2010). A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, (ACL)*, pages 1386–1395.

[67] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processingn, (EMNLP)*, pages 1532–1543.

[68] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, (NAACL): Human Language Technologies*, pages 2227–2237.

[69] Piktus, A., Edizel, N. B., Bojanowski, P., Grave, E., Ferreira, R., and Silvestri, F. (2019). Misspelling oblivious word embeddings. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, (NACACL): Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3226–3234.

[70] Rafae, A., Qayyum, A., Moeenuddin, M., Karim, A., Sajjad, H., and Kamiran, F. (2015). An unsupervised method for discovering lexical variations in roman urdu informal text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 823–828.

[71] Rashid, J., Shah, S. M. A., and Irtaza, A. (2019). Fuzzy topic modeling approach for text mining over short text. *Information Processing & Management, (IPM)*, 56(6):102060.

[72] Reimers, N. and Gurevych, I. (2017). Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, (EMNLP)*, pages 338–348.

[73] Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning, (ICML)*, pages 791–798.

[74] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

[75] Socher, R., Huang, E. H., Pennin, J., Manning, C. D., and Ng, A. Y. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the International Conference on Neural Information Processing Systems, (NIPS)*, pages 801–809.

[76] Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, (IPM)*, 45(4):427–437.

[77] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, (JMLR)*, 15(1):1929–1958.

[78] Subramani, S., Michalska, S., Wang, H., Du, J., Zhang, Y., and Shakeel, H. (2019). Deep learning for multi-class identification from domestic violence online posts. *IEEE Access*, 7:46210–46224.

[79] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems, (NIPS)*, pages 3104–3112.

[80] Tang, D., Qin, B., and Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, (EMNLP)*, pages 1422–1432.

[81] Tomar, G. S., Duque, T., Täckström, O., Uszkoreit, J., and Das, D. (2017). Neural paraphrase identification of questions with noisy pretraining. In *Proceedings of the Workshop on Subword and Character Level Models in NLP, (SCLeM)*, pages 142–147.

[82] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems, (NIPS)*, pages 5998–6008.

[83] Vila, M., Martí, M. A., and Rodríguez, H. (2014). Is this a paraphrase? what kind? paraphrase boundaries and typology. *Open Journal of Modern Linguistics, (OJML)*, 4(01):205–218.

[84] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning, (ICML)*, pages 1096–1103.

[85] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition, (CVPR)*, pages 3156–3164.

[86] Vo, N. P. A., Magnolini, S., and Popescu, O. (2015). Paraphrase identification and semantic similarity in twitter with simple features. In *Proceedings of the International Workshop on Natural Language Processing for Social Media, (Social NLP)*, pages 10–19.

[87] Wang, C., Duan, N., Zhou, M., and Zhang, M. (2013). Paraphrasing adaptation for web search ranking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, (ACL) (Short Papers)*, volume 2, pages 41–46.

[88] Wang, H., Can, D., Kazemzadeh, A., Bar, F., and Narayanan, S. (2012). A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the Association for Computational Linguistics System Demonstrations, (ACL-SD)*, pages 115–120.

[89] Wang, J. and Perez, L. (2017). The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks for Visual Recognition, (CNNVR)*, pages 1–8.

[90] Wang, X., Jiang, W., and Luo, Z. (2016a). Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of the International Conference on Computational Linguistics, (COLING): Technical Papers*, pages 2428–2437.

[91] Wang, Z., Hamza, W., and Florian, R. (2017). Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the International Joint Conference on Artificial Intelligence, (IJCAI)*, pages 4144–4150.

[92] Wang, Z. and Ittycheriah, A. (2015). Faq-based question answering via word alignment. *arXiv preprint arXiv:1507.02628*.

[93] Wang, Z., Lee, S., Li, S., and Zhou, G. (2015). Emotion detection in code-switching texts via bilingual and sentimental information. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 763–768.

[94] Wang, Z., Mi, H., and Ittycheriah, A. (2016b). Sentence similarity learning by lexical decomposition and composition. In *Proceedings of the International Conference on Computational Linguistics, (COLING)*, pages 1340–1349.

[95] Wang, Z., Zhang, Y., Lee, S., Li, S., and Zhou, G. (2016c). A bilingual attention network for code-switched emotion prediction. In *International Conference on Computational Linguistics (COLING): Technical Papers*, pages 1624–1634.

[96] Williams, A., Srinivasan, M., Liu, C., Lee, P., and Zhou, Q. (2019). Why do bilinguals code-switch when emotional? insights from immigrant parent-child interactions. *Emotion (Washington, DC)*.

[97] Xu, W., Callison-Burch, C., and Dolan, B. (2015). Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). In *Proceedings of the International Workshop on Semantic Evaluation, (SemEval)*, pages 1–11.

[98] Xu, W., Ritter, A., Callison-Burch, C., Dolan, W. B., and Ji, Y. (2014). Extracting lexically divergent paraphrases from twitter. *Transactions of the Association for Computational Linguistics, (TACL)*, 2:435–448.

[99] Yin, W., Schütze, H., Xiang, B., and Zhou, B. (2016). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics, (TACL)*, 4:259–272.

[100] Zarrella, G., Henderson, J., Merkhofer, E. M., and Strickhart, L. (2015). Mitre: Seven systems for semantic similarity in tweets. In *Proceedings of the International Workshop on Semantic Evaluation, (SemEval)*, pages 12–17.

[101] Zhao, J. and Lan, M. (2015). Ecnu: Leveraging word embeddings to boost performance for paraphrase in twitter. In *Proceedings of the International Workshop on Semantic Evaluation, (SemEval)*, pages 34–39.

[102] Zhou, X., Wan, X., and Xiao, J. (2016). Attention-based lstm network for cross-lingual sentiment classification. In *Conference on Empirical Methods in Natural Language Processing, (EMNLP)*, pages 247–256.

Appendix A

Effect of Paraphrase Transitive Extension Levels on SemEval Dataset

Table A.1 Paraphrase detection performance on SemEval dataset with K=1

Augmentation	Learned Features				Learned + Linguistic Features			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
P2, P3	82.8	57.9	65.1	61.3	86.8	71.3	61.1	65.8
P2, P3, P1	82.1	55.9	68.0	61.3	85.3	66.9	58.9	62.6
P2, P3, NP1	81.9	55.2	69.7	61.6	**87.4**	**72.5**	63.4	**67.7**
P2, P3, P1, NP1	82.8	57.1	**70.9**	63.3	86.8	72.2	59.4	65.2
P2, P3, P1, NP1, NP2	**85.2**	**64.9**	63.4	**64.2**	84.6	61.9	**68.6**	65.1

Table A.2 Paraphrase detection performance on SemEval dataset with K=2

Augmentation	Learned Features				Learned + Linguistic Features			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
P2, P3	82.2	56.7	62.9	59.6	85.0	65.6	58.9	62.0
P2, P3, P1	81.1	53.6	**72.0**	61.4	83.3	58.1	71.4	64.1
P2, P3, NP1	82.0	56.3	61.7	58.9	85.1	65.6	60.0	62.7
P2, P3, P1, NP1	83.5	59.9	64.0	61.9	**85.4**	**66.5**	61.1	63.7
P2, P3, P1, NP1, NP2	**84.2**	**62.1**	62.9	**62.5**	83.2	57.1	**78.3**	**66.0**

Table A.3 Paraphrase detection performance on SemEval dataset with K=3

Augmentation	Learned Features				Learned + Linguistic Features			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
P2, P3	82.6	57.9	60.6	59.2	86.0	63.7	61.1	62.4
P2, P3, P1	81.7	54.8	**71.4**	62.0	86.3	67.6	65.7	66.7
P2, P3, NP1	85.0	69.0	50.9	58.6	81.5	54.1	**75.4**	63.0
P2, P3, P1, NP1	**86.0**	**73.4**	52.0	60.9	**86.8**	**69.5**	65.1	**67.3**
P3, P2, P1, NP1, NP2	85.6	65.9	64.0	**64.9**	85.6	65.2	66.3	65.7