Towards a Deeper Understanding of Code-Mixing

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computational Linguistics by Research

by

Mrinal Dhar
201325118
mrinal.dhar@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
November 2018

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Towards a Deeper Understanding of Code-Mixing" by Mrinal Dhar, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. Manish Shrivastava

To my parents.

Acknowledgments

I don't even know who to thank. Even though there are people without whom this thesis would not have been possible, I cannot think of a single person to thank and acknowledge here.

I'm not going to thank my advisor, Prof. Manish Shrivastava, for being the coolest and the most understanding guide I could have asked for in this institute. It would not be an overstatement to say that this thesis exists because of him and his unwavering confidence in me.

Maybe I'll even forget thanking Prof. Dipti Mishra Sharma, for inspiring me from my very first day here five years ago. There has not been even one interaction between us in which I didn't get to learn something from her.

I certainly would not like to thank my parents and grandparents, who have never failed to inspire me with their work ethics, integrity, and more qualities than what can fit here. Without them, I'd be lost. My sister, Mansi, deserves a special mention for making my life so happy.

Vatika Harlalka, no thanks for making me do half the work I've done in college, including this thesis, by bugging me to do it until I did. I'd have given up a long time ago if not for you. Why should I thank Vaibhav Kumar, just for showing me what a good researcher looks like and trying to teach me the ways of being one, and Chanakya Malireddy, for being the best coding teammate while pulling those all-nighters on our projects?

I also shouldn't have to thank Mounika, Lasya, Sneha and Sriharsh for being there for me when I needed someone to talk to, even though you have had a profound impact on my life and I will always cherish our conversations. Yash, no thanks for those amazing memories from our work in San Francisco (and the party in Vegas!). I won't be thanking Mohanty "sir", for inspiring me and generations of students before me to aim so much higher than what others expected of us.

And finally, the only people who might ever actually read this page: Bhavesh, Chirag, Vaibhav, Sambuddha, Ishaan, Amogh, Kartik. They say a man is judged by the company he keeps. If that's true, I'm very happy with my chances. I won't thank you because one doesn't need to thank his brothers.

All jokes aside, I am eternally grateful to my advisor, my professors, family and friends, and this thesis is a culmination of the effort and support of these very important people in my life.

Thank you everyone.

Abstract

Over the last decade, there has been an exponential increase in the popularity of social media platforms such as Facebook, Twitter and other online social networking websites. Apart from offering a fast, easy and reliable method of communication with friends and family, for many people they have also become the primary source of information on current events and happenings around the world. With the rise of these social media networks, it has become extremely easy for anyone and everyone to share text, images and videos with other people. It has also proven to be a boon for creative people, by allowing them to share their work with a lot of people instantly at the press of a button. The advantages of these online social networks are manifold. Unfortunately, some characteristics of user generated content (UGC) poses problems in the way of natural language processing of the data that is shared by users on these websites. UGC typically contains non-standard spellings of words, unusual abbreviations, informal grammatical structure and variations in vocabulary, which are all issues that are not easily mitigated and therefore render current systems ineffective in processing such information.

In recent years, a phenomenon called "code-mixing", which is frequently observed in online social media, has attracted a lot of research and interest from sociolinguists. A sentence is said to be code-mixed if it consists of linguistic units, such as phrases, words or morphemes, from multiple languages. Code-mixing is often observed in the utterances of multilingual speakers, and is prevalent in countries where people speak multiple languages with native proficiency such as India.

The most prominent effect of code-mixing on everyday life can be seen in UGC on online social networks. Due to the informal nature of UGC, code-mixing is bound to happen especially in text written by multilingual writers. Since this hybrid code-mixed language does not have a formally defined grammar, it becomes a challenging task to process this information and analyze it further.

In this thesis, we attempt to understand the phenomenon of code-mixing in a deeper manner by analyzing text from online social media platforms. For deep learning NLP approaches to work on code-mixed text, they require an abundance of appropriate data to train on. Currently, there is a lack of such resources, therefore we start this work with the creation of 6096 English-Hindi code-mixed to monolingual English parallel corpus, which could aid deep learning methods in processing code-mixed text. We then proceed to develop a pipeline that can augment existing Machine Translation (MT) systems to enable support for translating code-mixed languages to monolingual. We present the results obtained by experiments conducted that augmented existing statistical and neural network based machine translation systems such as Google Translate, Bing and Moses. This pipeline was also used to

augment the classification of sentiments for code-mixed sentences by existing systems, by translating code-mixed English-Hindi sentences to English using the augmented MT systems and then using an available sentiment analysis system for English.

We take our work on code-mixing further by developing a hybrid architecture model that relies on neural networks with Attention mechanism as well as linguistic features to classify the sentiment labels for code-mixed sentences collected from online social networks such as Facebook and Twitter. We report these results which show that our approach is able to outperform previous methods for this task.

Finally, we demonstrate that a similar hybrid architecture can be used for other NLP tasks as well, such as Clickbait detection and Aggression identification of online social media posts. Our approach imporved on the results obtained by previous methods for the task of clickbait detection. The results of these experiments are also presented in this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's world, information is power. We live in a day and age where everyone is striving to possess and understand more and more information, countries and corporations alike. Computers have made this task easier and more efficient than ever, and we are consuming information at a higher rate than ever before. With the advent of computers came the Internet and it brought about a revolution in the way people communicated with each other, and the rest of the world.

In the last decade, digital communication mediums like e-mail, Facebook, Twitter, etc. have allowed people to have conversations in a much more informal manner than before. This informal nature of conversations has given rise to a new form of hybrid language, called *code-mixed* language, that lacks a formally defined structure.

1.1 Code-mixing

Myers-Scotton [41] defines code-mixing as "the embedding of linguistic units such as phrases, words and morphemes of one language into an utterance of another language". It is often used interchangeably with *code-switching*, which is a similar concept, except the fact that code-mixing is observed entirely in a single sentence, while code-switching occurs across sentences [25]. For the purposes of this study, we will not make a difference between code-mixing and code-switching and treat them both as code-mixing.

Code-mixing of Hindi and English, where sentences follow the syntax of Hindi but borrow some vocabulary from English is very prevalent on social media content in India, where most people are multi-lingual. An example of a code-mixed English-Hindi sentence is presented below:

- "*Main kal* movie *dekhne jaa rahi thi* and *raaste me* I met Sam."

  Gloss : I yesterday [movie] to-see go Continous-marker was [and] way in [I met Sam].

  English Translation : I was going for a movie yesterday and I met Sam on the way.

- *tu udhar ka* permanent *intezaam karke aa* !

  Gloss: you there is [permanent] arrangement do come !

  Translation: You come after making a permanent arrangement there !

- btw i was thinking *mai pehle ghar chale jaungi*, and *sham ko* venue *pe aa jaungi*

  Gloss: [btw i was thinking] i first home walk go, [and] evening [venue] on come go

  Translation: btw I was thinking that I'll first go home and come to venue in the evening.

This phenomenon is present in informal communication in almost every multi-lingual society, as studied by authors in [17]. They investigated how language used on these social media platforms, which they have called *texting* language, differs from the standard language that is found in more formal texts like books. The language used on these platforms resembles spoken conversations. It is also more common in the areas of the world where people are naturally bi- or multi-lingual. Usually, these are the areas where languages change over short geo-spatial distances and people generally have at least a basic knowledge of the neighbouring languages [29]. A very good example for this is a country like India which has an extensive language diversity and where dialectal changes frequently instigate code-mixing.

1.1.1 Challenges in Code-mixing

From the usage trend of Indians on online social media, it can easily be observed that the most commonly used language pair is English and Hindi. These languages are most popular among the general population of India which uses these social media websites.

However, Hindi and English are quite different languages, and therefore pose some interesting challenges when code-mixed, as described here:

- Words of Hindi language are written in Devanagari script in their standard form, while English uses the Roman script. As the majority of the websites on the Internet are in English, and perhaps because of the status it enjoys in India, code-mixed text on these websites is usually written in Roman script as well. When Hindi words are written in Roman script in a code-mixed sentence, one cannot be certain about their spellings, as it is a non-standard form of the words.

  For example, the Hindi word for sister, "बहन", can be written as "behen", "bahen", "bhen" in Roman script. There are multiple possible ways of writing the same Hindi word in Roman script because there is no definite one-to-one mapping between characters in Roman and Devanagari. This poses a problem of normalization, as different spellings of the same word must be normalized to be able to process it correctly.

- Hindi sentences typically follow Subject-Object-Verb word order, while English follows Subject-Verb-Object ordering. However, the resulting code-mixed language comprised of English and Hindi does not follow any such definitive ordering. This makes it harder to develop rules for the analysis of code-mixed sentences.

- As the code-mixed language does not have a formally defined grammar, it is difficult to develop rule-based systems for processing code-mixed data. Usually, the code-mixed language adopts the syntax of one of the languages which comprises it. This language thus lends its grammatical rules in which words are embedded into from the other code-mixed language.

  For example, the code-mixed sentence "I toh never went there kabhi bhi" follows the grammar of English while there are certain Hindi words like "toh", "kabhi" and "bhi" embedded in it. Identifying which language provides the grammar and which language is only lending vocabulary is a problem faced when dealing with code-mixed text.

1.2 Motivation for this study

In recent times, we have seen an explosion of Computer Mediated Communication (CMC) worldwide [27]. In CMC, language use lies somewhere in between spoken and written forms of a language and tend to use simple shorter constructions, contractions and phrasal repetitions typical of speech [19]. Such conversations, especially in social-media are both multi-party and multilingual, with mixing occurring between two or more languages, where the choice of language-use being highly influenced by the speakers and their communicative goals [18]. With multiple languages coming into play, and the variety of factors which influence the usage of those, the task of processing text becomes quite difficult.

As of now, according to the data of internetworldstats.com, the Internet has four-hundred fifty million English speaking users out of one billion five hundred million total users. This means that the market for English language is slightly less than one third of the total market. In other terms, most current approaches to information extraction exploiting social media and user-generated content (UGC), that are predominantly developed for English, are working with a mere third of the total data available.

The huge part of UGC that is not in English is currently being neglected mostly due to the relatively ephemeral character of UGC in general. Such content remains usually untranslated unless the users themselves so choose, because 1) it expresses opinions, and most users are much more likely to express or look for other opinions in the same language as their own rather than translated from a different one; 2) it is generated and updated at an extremely fast pace and has a very short lifespan, which rules out in practice the possibility of translation by human subjects and 3) is also produced in immense quantities, which, together with the previous point, ends up rendering translation by human subjects effectively impossible (both in terms of time and cost). All this leads to an enormous body of information being constantly generated which is also being constantly lost behind language barriers: the consolidation of Web 2.0 has caused an unprecedented increase in the amount of data and each individual user is currently being deprived of most of it [11].

These language barriers are intensified by the fact that a huge number of people do not use just one language, but multiple languages simultaneously, in the form of code-mixing. Even if we were able to create a system capable of processing information in every language, or perhaps a system capable

of translating text from any given language to another, we would still not be able to break down the language barrier completely, due to the phenomenon of code-mixing.

We are able to identify two kinds of consumers of code-mixed content: 1) Consumers who need to understand the information contained in the text; 2) Consumers who need not understand the information but need to be able to analyze it. For example, Machine Translation (MT) would help the first consumer to understand the complete meaning portrayed in the code-mixed content in a language that is understood by the consumer. On the other hand, tasks like Sentiment Analysis or Aggression identification operate on a higher level than MT by providing the consumer with an analysis of the information contained in the code-mixed content based on pre-defined metrics and evaluation criteria. It is crucial to develop strategies for the processing of code-mixed data for both kinds of consumers in order to develop a holistic solution for the understanding of code-mixed content.

1.3 Contributions of the Thesis

While code-mixing has been around for a while now, there is a huge difference in the efficiency of systems between processing standard languages and code-mixed languages. Due to the complications that arise when code-mixing is involved, the gap between available information and accessible information is growing day by day. In this thesis, we aim to take a few steps towards bridging that gap while keeping in mind both kinds of consumers as described above.

One of the most significant tasks in the processing of code-mixed language content is Machine Translation (MT). While translation of code-mixed text has been a requirement for some time, there is a noticeable lack of resources for this task. Without the availability of abundant resources, state-of-the-art deep learning approaches are unable to accurately translate code-mixed text. To tackle this problem, we have created a set of 6096 English-Hindi code-mixed and monolingual English gold standard parallel sentences. We started with publicly available datasets of code-mixed sentences collected from social media, and four annotators who were fluent in both English and Hindi, translated sentences from these datasets. We hope that this can serve as an initial attempt to promote generation of data resources for this domain.

Most MT systems, however, require a significant number of "parallel" sentences to perform well. While we wait for large parallel corpora for code-mixed text to be developed, we could make do by equipping the existing state-of-the-art MT systems to handle code-mixed content. This necessity has motivated us to develop an augmentation pipeline to support code-mixing on existing MT systems. Instead of developing special MT systems for code-mixed languages, we demonstrate how using our pipeline in conjunction with existing systems can produce better translations than those systems alone. Furthermore, this pipeline has been developed such that it is plugable with any existing MT system as a black-box entity, without any need for internal adjustments to be made to the pipeline.

Another important area of research in this domain is the sentiment analysis of code-mixed language content. With a huge number of people producing code-mixed content everyday, it is imperative, now

more than ever, to be able to decode the sentiment behind this content in order to understand popular trends, product reviews, favorability of a political candidate, etc. Owing to great variations in code-mixed content written by different people, standard rule-based approaches for sentiment analysis are unable to do an efficient job at this classification task, while their deep learning counterparts are limited by their dependence on the availability of data resources.

We propose a hybrid neural network framework, called *Code-Mixed Sentiment Analysis* or CMSA, that tackles the limitations of the previous systems by combining deep learning techniques with training of surface features and sentence vector representations, which we call a *Feature Network*. By augmenting the neural network with specific linguistic features, we are able to work around the problem of lack of abundant data. We developed a neural network architecture that utilizes Attention mechanisms with LSTMs and simultaneously learns a Feature Network, using it to augment the deep learning techniques for the task of sentiment analysis. In addition, we performed extensive experimental evaluation to demonstrate the effectiveness of our system in comparison to state-of-the-art and traditional techniques for the sentiment analysis task. To the best of our knowledge, our system outperforms the state-of-the-art systems for sentiment analysis of English-Hindi code-mixed data. We also explore how the augmentation pipeline we developed for improving machine translation of code-mixed data can be used to help with the sentiment analysis task. We report the results of the experiments performed by translating the code-mixed data to English first, and then applying state-of-the-art sentiment analysis tools to perform a sentiment classification.

Using a similar hybrid architecture consisting of neural networks and surface features, we also developed an approach for detecting "clickbait" in online articles. Clickbait, which is used to entice users into clicking a link for an article by piquing their curiosity under false pretenses, is used indiscriminately by media outlets on the Internet in an effort to maximize their revenue generated by advertisements. We explore how our hybrid model can be used to detect such article titles, in an effort to reduce the spread of such practices.

1.4 Thesis Organization

This thesis has been divided into 6 chapters. We present a detailed study of existing research in this field and how it relates to our work in Chapter 2.

In Chapter 3, we describe the English-Hindi code-mixed to monolingual English parallel corpus that we created, along with observations and our methodology for annotation. We also talk about the augmentation pipeline that we developed for existing machine translation systems in detail, along with a detailed report of the experiments conducted to verify that augmentation yields better results.

Chapter 4 deals with the task of Sentiment Analysis of code-mixed English-Hindi sentences, and presents our hybrid architecture of a combination of neural networks with attention mechanism and linguistic features.

This is followed by Chapter 5, a discussion of other applications for this hybrid architecture, including its use for the task of Clickbait Detection on online social media data. Finally, we conclude our thesis and discuss the scope of future work in Chapter 6.

Chapter 2

Related Work

In recent times, there has been a lot of interest from the Computational Linguistics community to support code-mixing in language systems and models. Machine Translation and Sentiment Analysis are both considered reasonably advanced in terms of available research, and these tasks have been worked on for decades now. However, when the case of code-mixed language content is considered, research in the domain of Machine Translation and Sentiment Analysis has only gained interest in the last few years.

2.1 Code-mixing in Social Media

In [61], the authors performed some of the earliest work in the analysis of the effects of code-switching in online communications. By conducting linguistic analysis, surveys and interviews, they explored how English and Arabic use varied online in Egypt. They found that the use of English was much more than Arabic in online and formal communications, and they analyzed this trend further in the context of technology and globalization.

Due to a massive growth of social media content, the usage of noisy non-standard tokens online has also increased. Hence, text normalization systems have become necessary that can convert these non-standard tokens to their standard form. Language identification at the word level was attempted by researchers in [43] on Turkish-Dutch posts collected from an online chat forum. They performed comparisons between different approaches such as using a dictionary and statistical models. They were able to develop a system which had an accuracy of 97.6%, and concluded that language models prove to be better than a dictionary based approach. In [6], the authors explored the same task on social media text in code-mixed Bengali-Hindi-English languages. They annotated a corpus with over 180,000 tokens, and used statistical models with monolingual dictionaries to achieve an accuracy of 95.76%.

The researchers in [59] deal with POS tagging of English-Hindi code-mixed data that they extracted from Twitter and Facebook. Social media is a good source for obtaining code-mixed data as it is the preferred choice of the urban youth as informal platforms for communication. In [5], the authors have done a study of English-Hindi code-mixing on Facebook, and their investigation demonstrates the extent

of code-mixing in the digital world, and the need for systems that can automatically process this data. The authors in [55] have addressed shallow parsing of English-Hindi code-mixed data obtained from online social media. They were the first to attempt shallow parsing of code-mixed data, to the best of our knowledge.

Language identification in code-mixed data is an important step because it might determine how to further process the data in tasks like POS tagging, for example. In [14], researchers explore a CRF based system for word level identification of languages. Using lexical, contextual, character n-gram and special character features, they were able to replicate performance of their system on four language pairs: English-Spanish (En-Es), English-Nepali (En-Ne), English-Mandarin (En-Cn), and Standard Arabic-Arabic (Ar-Ar) Dialects.

Apart from this, the authors in [51] have explored the problem of classification of code-mixed questions. By translating words to English before extracting features, they were able to leverage the resources available in a resource-rich language. Using word level resources with Support Vector Machines (SVM), they were able to achieve an accuracy of 63%. WebShodh, developed by the researchers in [13], is an end-to-end open domain factoid, web-based question answering system that is based on this work. It provides ranked recommendations as potential answers to a code-mixed question. They were the first to develop such a system specifically for code-mixed questions.

2.2 Machine Translation

Machine Translation of code-mixed language content has received a lot of attention in recent time. With the proliferation of user generated content, there is need to translate this data so that people who speak other languages can also understand it. Developing translation systems specifically for code-mixed language usually requires an abundance of linguistic resources that can be used by such systems. Although currently there are no such publicly available parallel corpus, efforts are being made in that direction.

In [26], the authors created a dataset of 1446 code-mixed English-Hindi sentences with the associated language identification and normalization. This was the first attempt to create such a linguistic resource for the language pair. They also presented an empirical study detailing the construction of language identification and normalization system designed for the language pair. The researchers in [30] also released a dataset of 3879 code-mixed English-Hindi sentences with their associated sentiment labels.

Code-mixed translation has been attempted previously by the authors in [57] from a linguistics perspective. They analyze code-mixing in the English-Hindi language pair, and identify word features from both the languages using a technique they call cross morphological analysis. However, we primarily draw our inspiration from the skeleton model presented by the authors in [53]. The broad idea they present is similar to ours, but we found that many of the assumptions made were very simplistic and no evaluation or results were provided as well as no systems were released. We provide a deeper look into

the code-mixed translation process and demonstrate the impact on existing machine translation systems such as Google Translate, Microsoft Bing and Moses, along with the release of a benchmark dataset.

2.3 Sentiment Analysis

The problem of sentiment analysis has attracted extensive studies and a myriad of publications. In [40], the researchers created two SVM based classifiers to detect sentiments in tweets and text messages respectively, involving the use of a number of surface-form and semantic features.

In [9], the authors proposed a skip-gram based word representation model that proved to be effective for languages with large vocabularies, and could be useful in classification tasks such as sentiment analysis. The researchers in [23] trained document vectors lexicon-based, word embedding-based and hybrid vectorizations with a polarity classification model for sentiment analysis. Their approach proves to be useful when there is a lack of available computational resources.

Another approach has been to use ensemble techniques for improving sentiment analysis by deep learning systems [3]. They used word embeddings and a linear machine learning algorithm as their base classifier and aggregate it with two models for surface classifiers used for sentiment analysis.

There is some work for Code-mixed sentiment analysis too. In [56], the authors presented an approach based on lexicon lookup for text normalization and sentiment analysis of code-mixed data. The researchers in [30] used sub-word level compositions to learn information about the sentiment value at the morpheme level. The authors in [16] use siamese networks to maintain a common sentiment space mapping between code-mixed and standard languages. They utilize contrastive learning to classify sentiments in code-mixed content.

Although, the above models work well, they suffer from the lack of abundant data required for training a good deep model. We show that by looking at the input differently and by augmenting the model with specific features we can achieve significant improvements on the state-of-the-art.

2.4 Clickbait Detection

There has been a paradigm shift in the preference of people in the medium of consuming news and other articles. From print media, the bias has now changed towards news sources on the Internet and media houses have not failed to capitalize on this changing trend. Given the rise in online targeted advertising as a major source of income for these media companies producing the news, the importance of detecting clickbait headlines has also increased exponentially in recent years. Clickbait headlines generate revenue by appealing to the reader's curiosity. They provide incomplete information that attracts the user to click on them to learn more about something, which often turns out to be fake or unimportant.

Initial work in this domain can be traced back to the researchers in [8], relying on heavy feature engineering on a specific news dataset. These works define the various types of clickbait and focus on the presence of linguistic peculiarities in the headline text, including various informality metrics and

the use of forward references. Applying such techniques over a social media stream was first attempted by the authors in [48] as the authors crowdsourced a dataset of tweets [47] and performed feature engineering to accomplish the task. In [12], researchers have tried to expand the work done for news headlines they collected from trusted sources.

The authors in [2] used the same collection of headlines as the authors in [12] and proposed the first neural network based approach in the field. They employed various recurrent neural network architectures to model sequential data and its dependencies, taking as its inputs a concatenation of the word and character-level embeddings of the headline. Their experiments yielded that bidirectional LSTMs [54] were best suited for the same. In [58], the researchers built BiLSTMs to model each textual attribute of the post (post-text, target-title, target-paragraphs, target-description, target-keywords, post-time) available in the corpus [47], concatenating their outputs and feeding it to a fully connected layer to classify the post. Attention mechanisms [4] have grown popular for various text classification tasks, like aspect based sentiment analysis. Utilizing this technique, the researchers in [65] deployed a self-attentive bidirectional GRU to infer the importance of each tweet token and model the annotation distribution of headlines in the corpus.

Word vectors and character vectors have been used across various approaches proposed to solve this problem. However, we suggest the use of subword representations to better analyse the morphology of possible clickbait-y words. We also attempt to model the interaction between the title of an article and its text.

Chapter 3

Enabling Code-Mixed Machine Translation

In the last decade, digital communication mediums like e-mail, Facebook, Twitter, etc. have allowed people to have conversations in a much more informal manner than before. This informal nature of conversations has given rise to a new form of hybrid language, called *code-mixed* language, that lacks a formally defined structure.

With the meteoric rise in the quantity of code-mixed user generated content (UGC) produced every day, a need has arisen for Machine Translation (MT) of such data into other languages such that users from across the globe can understand it and be a part of the discussion. However, current state-of-the-art approaches for MT are limited due to their dependence on abundant linguistic resources, which are not available for code-mixed content. This limitation prohibits deep learning based approaches from generating accurate translations.

While translation of code-mixed text has been a requirement for some time, there is a noticeable lack of resources for this task. In an effort to tackle this problem, we created and released a set of 6096 English-Hindi code-mixed and monolingual English gold standard parallel sentences. We started by collecting publicly available datasets of code-mixed sentences collected from social media, and four annotators who were fluent in both English and Hindi then translated these sentences. In the following sections, we describe the motivation behind this task, along with the corpus creation process and our analysis of the datasets collected and used for the same.

With the help of the created parallel corpus, we analyzed the structure of English-Hindi code-mixed data and present a technique to augment run-of-the-mill machine translation (MT) approaches that can help achieve superior translations without the need for specially designed translation systems. We present an augmentation pipeline for existing MT approaches, like Phrase Based MT (Moses) and Neural MT, to improve the translation of code-mixed text. The augmentation pipeline is presented as a pre-processing step and can be plugged with any existing MT system, which we demonstrate by improving translations done by systems like Moses, Google Neural Machine Translation System (NMTS) and Bing Translator for English-Hindi code-mixed content.

3.1 Motivation

The Information Age brought many challenges along with it. In order to ease information exchange between people of different ethnicity, varying backgrounds and cultures, there is a need to break down the language barrier between them. In a highly populated country full of diverse languages like India, phenomenon such as code-mixing is very common. Code-Mixing is a very frequently observed phenomenon in social media content generated by multilingual speakers. The processing of code-mixed data for computational modeling is very challenging due to the linguistic complexity resulting from the nature of the mixing. Therefore, seamless exchange of information can only be done by the use of a system that accurately translates between code-mixed languages into the language of choice.

Recently, tasks such as shallow parsing has been accomplished over code-mixed data, however, little attention has been paid to Machine Translation of such data. Furthermore, no parallel corpus of code-mixed English-Hindi and English exists. As a result, even state-of-the-art systems in Machine Translation, which are typically based on deep learning approaches, are limited by the lack of abundant data which is required by such approaches.

We present a newly created parallel corpus of code-mixed English-Hindi. We selected previously available English-Hindi code-mixed data as a starting point for the creation of our parallel corpus. In some of the data available to us, tokens in the sentences were already normalized. We chose 4 annotators fluent in speaking both English and Hindi. We first annotated 100 sentences and calculated agreement between them. After that we came up with suitable guidelines in case of conflicts. We finally translated 6096 English-Hindi code-mixed sentences to English as well as Hindi.

With this corpus, we aim to take a step towards advancing the task of code-mixed translation by facilitating data-driven research in this domain. However, most MT systems require a significant number of "parallel" sentences to perform well. While we wait for large parallel corpora for code-mixed text to be developed, we could make do by equipping the existing state-of-the-art MT systems to handle code-mixed content. This necessity has motivated us to develop an augmentation pipeline to support code-mixing on existing MT systems.

3.2 Creation of Parallel Corpus

3.2.1 Data collection

We started with the code-mixed dataset created and released by the authors in [26]. They collected 1446 sentences from social media, and performed language identification and word normalization on these sentences. Most of this data consisted of informal conversations between people, and therefore had quite a few misspellings that had to be resolved first.

We also obtained 764 English-Hindi code-mixed sentences from the dataset released as part of an NLP tools contest at ICON 2017 [1], created by collection of Whatsapp chat messages. Additionally, we use a dataset released by the authors in [30] for the task of sentiment analysis of code-mixed content,

which contains 3879 English-Hindi code-mixed sentences. They collected this data by scraping the comments from the Facebook pages of Indian celebrities.

For our study, we removed annotations such as sentiment labels, POS tags, etc. from the obtained datasets, and only used raw sentences for the task of corpus creation.

3.2.2 Data Statistics and Analysis

The 6096 code-mixed sentences contain a total of 63913 tokens. Of these tokens, 37673 are Hindi words and 16182 are English words. The rest of the tokens were marked as "Rest". "Rest" would mean that these tokens could be abbreviations, named entities etc. The tokens in the data were already normalized.

It is very crucial to find the level of code-mixing in data, since if the extent of code mixing is too less, then it is as good as a monolingual corpus and would not provide us much benefit in the task of code-mixed translation. Hence, in order to find the level of code-mixing present in the data, we use the Code-Mixing Index (CMI)[20] defined as:

$$CMI = \begin{cases} 100 \times [1 - \frac{max\{w_i\}}{n-u}] & n > u \\ 0 & n = u \end{cases} \tag{3.1}$$

where, w_i is the number of words tagged with a particular language tag, $max\{w_i\}$ represents the number of words of the most prominent language, n is the total number of tokens, u represents the number of language independent tokens (such as named entities, abbreviations), in our case these would be the words marked as 'Rest'.

Summarized statistics of the data can be found in Table 3.1. From the table we can see that the code-mixing index is around 30.5, hence, we can safely assume that the data has a decent variety of code-mixed sentences which could be effectively used for the creation of translation systems.

3.2.3 Methodology for Annotation

For the task of translation, we selected 4 annotators who were fluent in both English and Hindi. Before starting the process of annotation, we randomly sampled 100 sentences from the corpus. We then asked one of the annotators to translate the given sentences into English. The other three annotators judged the translated sentences into two categories, *Totally Correct (TC)* and *Requires Changes (RC)*. Finally, we use the Fleiss' Kappa measure in order to calculate agreement.

Fleiss' Kappa is a statistical measure for assessing the reliability of agreement between a fixed number of raters when assigning categorical ratings to a number of items or classifying items in order to calculate the agreement. The measure is defined as follows:

Let N denote the number of subjects, n denote the number of ratings per subject, and k be the number of categories into which assignments are to be made. In our case N=100, n=3, k=2. Let the

subjects be indexed by i = 1, ... N, categories be indexed by j = 1, ... k and n_{ij} be the number of raters who assigned the j^{th} category to the i^{th} subject. Then,

$$P_i = \frac{1}{n(n-1)}[(\sum_{j=1}^{k} {n_{ij}}^2) - (n)] \tag{3.2}$$

here, P_i denotes the extent to which raters agree to the i^{th} subject,

$$\overline{P} = \frac{1}{N}\sum_{i=1}^{N} P_i \tag{3.3}$$

$$p_j = \frac{1}{Nn}\sum_{i=1}^{N} n_{ij} \tag{3.4}$$

$$\overline{P_e} = \sum_{j=1}^{k} {p_j}^2 \tag{3.5}$$

and finally,

$$\kappa = \frac{\overline{P} - \overline{P_e}}{1 - \overline{P_e}} \tag{3.6}$$

where κ denotes the Fleiss' Kappa score.

Using this, we found out that the κ score for both the cases, i.e code-mixed to English as well as code-mixed to Hindi was close to 0.88. We speculate that such a high agreement could have arised from the fact that the annotators had the same cultural background and shared similar communities.

In Table 3.2 we provide a few examples of code-mixed English-Hindi translated to English with the help of our pipeline. From the examples we can clearly see that the dataset sentences consist of transliterated Hindi words. Transliterated words usually pose a problem because one needs to come up with a standard form of those words before proceeding to the task of translation. However, in [26], the authors had already normalized the transliterated words. Hence, we were able to translate sentences without having to normalize the words.

Type	**Value**
Total no. of code-mixed sentences	6096
Total no. of tokens	63913
Total no. of Hindi tokens	37673
Total no. of English tokens	16182
Total no. of 'Rest'	10094
Code Mixing Index	30.5

Table 3.1: Corpus Statistics

Some examples from the dataset which illustrate code-mixing:

1. I was really trying *ki aajayun*

   Gloss: [I was really trying] I come

   Translation: I was really trying that I come

2. Sorrrry, *aaj subah tak pata nhi tha* that I wudnt be able to come today

   Gloss: [Sorrrry], today morning till know not did [that I wudnt be able to come today]

   Translation: Sorrrry, I didn't know until today morning that I wouldn't be able to come today.

3. *tu udhar ka* permanent *intezaam karke aa* !

   Gloss: you there is [permanent] arrangement do come !

   Translation: You come after making a permanent arrangement there !

4. btw i was thinking *mai pehle ghar chale jaungi*, and *sham ko* venue *pe aa jaungi*

   Gloss: [btw i was thinking] i first home walk go, [and] evening [venue] on come go

   Translation: btw I was thinking that I'll first go home and come to venue in the evening.

3.2.4 Challenges faced

Due to the nature of code-mixed text, there are a number of complications that arose when translating between code-mixed and monolingual languages.

For example, a number of character pairs in English and Hindi do not have a one-to-one mapping, which results in ambiguity while identifying the correct meaning of the word. We relied on the context of the words to perform the translation correctly in this case.

Another complication was caused by the use of metaphors and typical Hindi sayings in the code-mixed data, which can either be translated literally or according to their "sense" in English. In order to obtain neutral and more standardized translations, we decided to interpret such sentences literally and translated them as such.

Finally, as social media data is very informal, in many cases there is a lack of punctuation, which when combined with the long sentence length make it much more difficult to perform the translations correctly. To get around this complication, we decided to add punctuation according to our interpretation of the source sentence. This ensured that the translations were legible and coherent.

3.3 MT Augmentation Pipeline

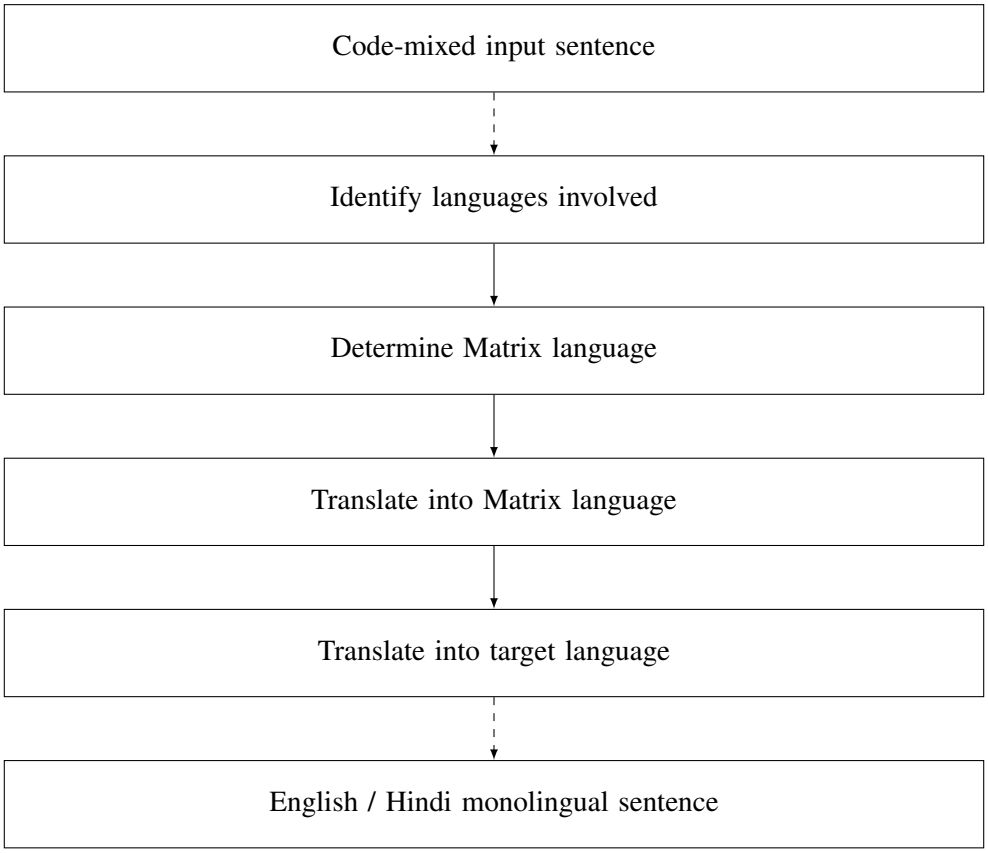

Figure 3.1: Translation augmentation pipeline

3.3.1 Approach

Instead of attempting to create a new machine translation system specifically for code-mixed text, which is not feasible due to lack of abundant gold standard data, we now introduce an approach that will augment existing systems for use with code-mixed languages in the form of the pipeline described in Figure 3.1.

In order to improve the results of the MT system for code-mixing, we attempt to change the input code-mixed sentence to more closely resemble the kind of input these systems work best for - a monolingual sentence. As the code-mixed sentence follows the structure of the Matrix language, we translate words of the code-mixed sentence that belong to the Embedded language (Em) to the Matrix (Mat) language using a machine translation system (Em-Mat MT). The resulting sentence would be translated to the desired target language (Tgt) using another existing MT system (Mat-Tgt MT), which may or may not be the same as the Em-Mat MT system (depending on what the matrix and target languages are). For

our experiments, we have fixed the target (Tgt) to be English, as we attempt to translate English-Hindi code-mixed language to English.

We now discuss each step of the augmentation pipeline in more detail.

3.3.1.1 Identification of languages involved

As a pre-processing step, we identify the languages that are involved in creating the hybrid code-mixed language in the text. This step is crucial in determining the pipeline to follow for the rest of the translation.

In [26], the authors released their dataset with manually annotated language identifiers associated with every word of the sentences in the data. For the other datasets, we make use of the Language Identification (LID) system by the researchers in [7] to identify the corresponding language for each word in a code-mixed sentence.

3.3.1.2 Determination of Matrix Language

In the Matrix Language-Frame model proposed by the authors in [42], a code-mixed sentence is formed by a Matrix language and an Embedded language. The overall morpho-syntactic structure of the sentence is that of the Matrix language, however, words from the Embedded language are also present in the sentence.

In other words, the Matrix language lends its syntactical structure and the Embedded language lends its vocabulary to the code-mixed sentence. The Matrix language is determined by employing the following heuristics in decreasing order of preference:

1. **The number of words in one language in the sentence**
   The language which has more words in the sentence is likely to be the matrix language.

2. **Determination of the syntactic structure of the text by detecting the language of the verb**
   For example, if the two languages involved in code-mixing are English and Hindi, and a sentence follows SVO structure, in that case English is likely to be the Matrix language of the sentence.

3. **Usage of function words of a particular language in the text**
   The language whose function words exist in the sentence might be the matrix language, since function words are associated with syntax.

3.3.1.3 Translation into Matrix Language

While most translation systems are unable to translate foreign (including code-mixed) words due to lack of training, borrowed and commonly used words are often found in parallel corpora. Commercial systems, like Google Translate, can translate frequent short phrases as well. Errors creep in when they have to deal with longer phrases. Keeping this in mind and to reduce the translation cost per sentence,

we selected **the longest string of words belonging to the Embedded language** for translation to the Matrix language. This is to ensure the maximal meaningful translation with a single call to the Em-Mat MT engine. This results in a code-mixed sentence that follows the syntax of the Matrix language and also has the majority of its words in this Matrix language.

Note that even though we only perform one translation at this step, that is, only for the longest string of words belonging to the Embedded language, we can also perform multiple translations by selecting all such strings one by one. However, that would considerably increase the time required to translate each sentence and thus may not be a good method to apply in real-world applications.

3.3.1.4 Translation into Target Language

Now that the code-mixed sentence has been translated in to the matrix language, it can be directly translated to the Target language using the Mat-Tgt MT engine.

The system recognizes when the target language is the same as the matrix language, and therefore there is no need to perform a translation of parts or whole of the sentence.

3.3.2 Evaluation and Results

In order to evaluate our methods of augmentation, we consider the following existing machine translation systems: **Moses** [34], **Google's Neural Machine Translation System (NMTS)** [62], **Bing Translator**.

For training a translation model for Moses, we used the English-Hindi parallel corpus released by the researchers in [36]. This dataset consists of 1,492,827 parallel monolingual English and monolingual Hindi sentences. The Hindi sentences were in the Devanagari script, and required pre-processing for use with our code-mixed dataset, which is entirely in Roman script. We use the aforementioned three MT systems for translating sentences from the code-mixed English-Hindi to monolingual English parallel corpus that we created (Section 3.2), with and without the use of our augmentation pipeline in order to draw a comparison.

From Table 3.2, the effects of using our augmentation pipeline can be seen on sentences from our English-Hindi code-mixed dataset. Note that without augmentation, Google translate is unable to consider words from Hindi embedded in the code-mixed sentence in the context of other English words. For example, in the sentence "tu udhar ka permanent intezaam karke aa!", the hindi words *intezaam karke* are not properly translated because the context of those is not being considered. In some another examples, the MT system completely ignored the Hindi words and did not perform any translation at all. Our pipeline is able to resolve these issues and provide a more accurate translation of code-mixed sentences.

Original sentence	**Without Augmentation**	**With Augmentation**
room *mei shayad kal bhi nahi* stay *karungi* , cancel *ho sakti hai uski* booking *abhi* ?	I will not stay in the room tomorrow, can I cancel her booking now?	I will not stay in the room tomorrow, can I cancel her booking now?
Sorrrry , *aaj subah tak pata nhi tha* that I wudnt be able to come today	Sorry , *aaj subah tak pata nahi tha* that I wouldn't be able to come today	Sorry , Did not know until this morning that I wudnt be able to come today
I was really trying *ki aajayun*	I was really trying *ki aajayun*	I was really trying I come
par if its possible and any other guest needs a room , *mera* room *de de kisi ko bhi*	*par* if its possible and any other guest needs a room , *mera* room *de de kisi ko bhi*	*par* if its possible and any other guest needs a room , Give my room to anyone
toh hum aaj train ki ticket karwa lenge .	So we will get a train ticket today.	So we will get a train ticket today.
tu udhar ka permanent *intezaam karke aa* !	You come here by arranging Permanent!	You come here with a permanent arrangement!

Table 3.2: Augmenting Google Translate with our pipeline

We compare the output translations of these MT systems for code-mixed data, with and without the augmentation by our system. For accuracy metrics, we chose BLEU score, Word Error Rate (WER) and Translation Error Rate (TER) as they are ideal for use with machine translation.

As can be observed from Table 3.3, our augmentation pipeline significantly improves the translation accuracy of existing machine translation systems. Note that among the systems themselves, Google NMTS performs much better on code-mixed English-Hindi data as compared to traditional phrase based systems like Moses and even neural systems like Bing Translator. Even though Moses does not perform as well as the other systems described here for code-mixed data, which can be attributed to the limited training data available for its model, our pipeline is still able to boost its performance significantly.

Note that even though we have demonstrated our pipeline by using a single MT system for both the translations (Em-Mat and Mat-Tgt), due to the modular nature of the pipeline, it can be used with any two machine translation systems.

	Without Augmentation			With Augmentation		
	BLEU	**WER**	**TER**	**BLEU**	**WER**	**TER**
Moses	14.9	10.671	2.403	16.9	9.505	2.295
Google NMTS	28.4	5.882	0.692	37.8	4.030	0.537
Bing Translator	18.9	8.940	1.108	25.0	8.054	0.917

Table 3.3: Comparison of performance with and without using our augmentation pipeline. (Note: BLEU- higher is better, (WER,TER)- lower is better.)

3.4 Conclusion

In summary, we have created and released a gold standard corpus consisting of 6096 code-mixed English-Hindi and monolingual English parallel sentences. The original sentences were obtained from various publicly available datasets, compiled from social media text. We also discussed the issues encountered during the translation process, caused due to the nature of code-mixed text, and our methods for overcoming them.

In addition to the corpus, we have developed a pre-processing pipeline, that can be used to augment existing machine translation systems such that translation of code-mixed data can be improved without training an MT system specifically for code-mixed text. Using the evaluation metrics selected, it is shown that there is a quantifiable improvement in the accuracy of translations with the augmentation proposed.

As part of our study, we have observed that long distance re-ordering of words is still an issue with code-mixed MT. Also, since code-mixed language does not have a standard form, it is difficult to establish a correct version of spelling for a particular word. Language identification is the most critical module for translation augmentation, because the same orthographic form can lead to valid words in multiple languages.

To take this work further, we intend to develop an end-to-end code-mixed MT system which can jointly perform the normalization, language identification, matrix language identification and two-step translation tasks. A hybrid model, partially trained on gold parallel corpus, may also be attempted.

Chapter 4

Sentiment Analysis of Code-Mixed Content

Sentiment analysis of online social media has many applications in e-commerce, recommendation systems, analysis of current trends, political campaigns, etc. In a multilingual society, such content is often a composition of different languages. This phenomenon of mixing the vocabulary and syntax of two or more languages (code-mixing) makes the processing of such content significantly harder.

We present a hybrid architecture for the task of Sentiment Analysis of English-Hindi code-mixed data. We use two different Bidirectional Long Short Term Memory (BiLSTM) Networks, one that looks at the overall sentiment of the sentence while the other utilizes an attention mechanism in order to focus on the individual sentiment bearing sub-words. This, combined with traditionally used orthographic features and monolingually trained word embeddings achieves the state-of-the-art results on a benchmark dataset. Our system scores an accuracy of 83.54% and an F1-score of 0.827.

4.1 Motivation

Sentiment analysis (SA) has a variety of applications in analyzing movie reviews, user modeling, curating online trends and political campaigns and opinion mining. A majority of this information comes from online social media such as Facebook and Twitter. In recent times, these websites have taken over traditional forms of digital communication like e-mail. As communication became more informal and as the ease of putting one's thoughts on the Internet increased, so did the presence of code-mixing online.

A large number of Indian users on online social media websites can speak English and Hindi with bilingual proficiency. Consequently, English-Hindi code-mixed content has become ubiquitous on the Internet which has created the need to process this form of natural language. Myers-Scotton [41] defines code-mixing as "the embedding of linguistic units such as phrases, words and morphemes of one language into an utterance of another language". Typically, a code-mixed sentence retains the underlying grammar and script of one of the languages it is comprised of.

However, given that there is no formally defined grammar for a code-mixed hybrid language, traditional approaches to SA do not work very well on code-mixed content. In [17], the authors investigated

how language used on these social media platforms, which they have called *texting* language, differs from the standard language that is found in more formal texts like books. Adding to the problem is the lack of available annotated code-mixed data for Sentiment Analysis, which is a severe limitation in itself for deep learning techniques.

Due to the nature of code-mixing, words from one of the two code-mixed languages may not be written in the same script as their standard form. For example, in the code-mixed sentence *bahan, where are you?*, the word *bahan* is written in a non-standard Roman spelling for the actual Hindi word "बहन", which means "sister". As there is no formally "correct" way of spelling these words in a non-standard script, this results in spelling variations between different writers and makes existing sentiment analysis systems unsuitable for code-mixed data. Therefore, there is a need to develop new systems that target code-mixed content specifically.

There have been multiple attempts in the past to develop solutions for this problem, including a shared task in ICON 2017 that targets Sentiment Analysis of code-mixed Indian languages specifically [45].

The authors in [50] make use of a lexicon lookup approach for performing domain specific sentiment analysis. Other attempts include using Sub-word level compositions with LSTMs to capture sentiment at morpheme level [30], or using contrastive learning with siamese networks to map code-mixed and standard language text to a common sentiment space [16]. While these systems are effective to a degree, they are primarily deep learning techniques and hence they're limited by their dependence on abundant data.

We propose a hybrid neural network framework, called *Code-Mixed Sentiment Analysis* or CMSA, that tackles the limitations of the previous systems by combining deep learning techniques with training of surface features and sentence vector representations, which we call a *Feature Network*. By augmenting the neural network with specific linguistic features, we are able to work around the lack of abundant data.

4.2 Model Architecture

In this section we present the architecture of our system. We first present the overview followed by the explanations of the various components of the model.

4.2.1 Overview

Our system consists of three primary components. The first component uses sub-word level representations with a Convolutional Neural Network (CNN). Sub-words provide the optimal level of detail in order for the CNN to capture morpheme level sentiment information. In Section 4.2.3, we describe our second component, a parallel encoder network consisting of two bi-directional Long Short Term

Memory (BiLSTM) Networks, that capture 1) sentiment information from specific parts of the sentence and 2) the overall sentiment information of a sentence.

Finally, the third component, which is a Feature Network, described in Section 4.2.4, consists of surface features and a vector representation of the sentence that can represent out-of-vocabulary words as well. It augments the neural network framework of our system to boost the classification accuracy for the task of sentiment analysis.

4.2.2 Sub-word level representations with CNN

We take inspiration from the work done by the authors in [30] involving sub-word level compositions for a similar task in sentiment analysis.

Traditionally, statistical approaches to sentiment analysis work on word-level feature representations. However, they make the assumption that a language has finite vocabulary which might work for English but clearly does not work in our case, given the problem of spelling variations in code-mixed language. Therefore, word-level representations would be too coarse for our task.

On the other end of the spectrum would be character-level feature representations. However, it is difficult to establish a mapping between the meaning of a word and the way it is constructed by characters. As one character inherently does not provide any semantic information that can be used for our purpose, we dismiss character-level feature representations as our choice of embeddings as well.

As a middle ground between the two discussed strategies, sub-word level embeddings prove to be an appropriate choice for representations for our task. While individual characters might not provide any semantic value on their own, groups of characters can hold meaning that can be interpreted by humans. For example: The word *disengage* can be broken down into the sub-words *dis* and *engage*, both of which project semantic information.

Therefore, we believe that sub-word level representations are correctly suited for our task.

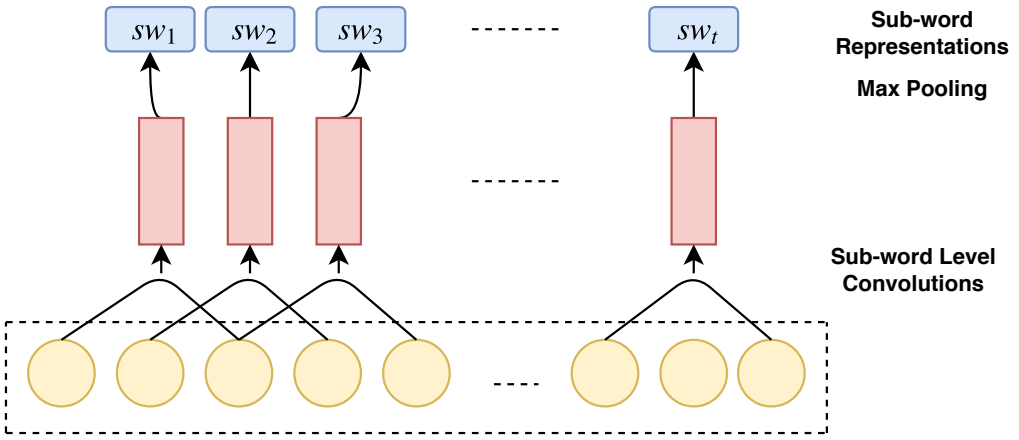

Figure 4.1: Convolution Neural Network for Generating Sub-Word Representation

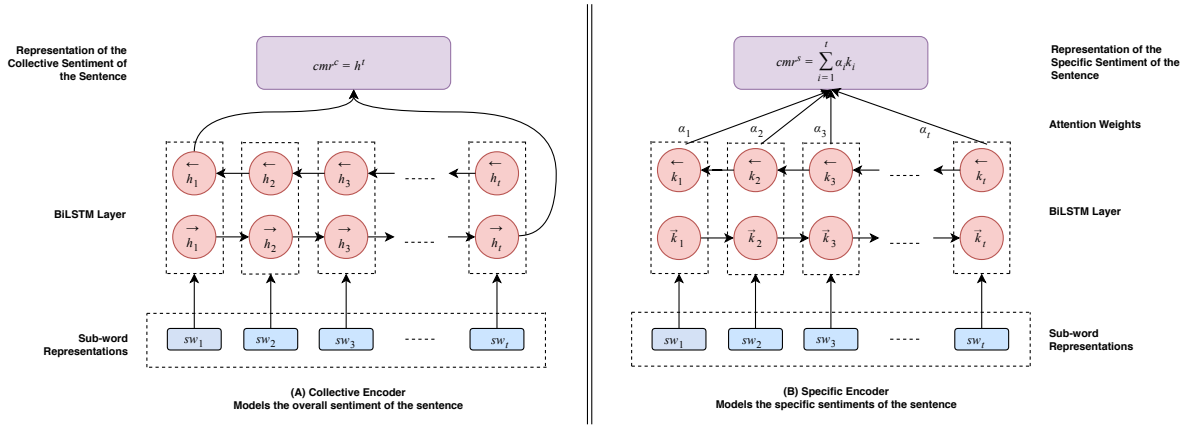

Figure 4.2: The two different encoders used in the model.

4.2.3 BiLSTM network with Attention mechanisms

We utilize a combination of two different encoders in our model, an overview of which was presented in Section 4.2.1. Below we explain both these encoders in detail.

4.2.3.1 Collective Encoder

The purpose of the collective encoder is to learn a representation which encapsulates the overall sentiment of the sentence. The graphical representation of this encoder can be seen in Fig. 4.2(A). The inputs to the Collective Encoder are sub-word representations obtained from the CNN. In this encoder, we use RNNs with BiLSTM cells. This encoder is useful in coming up with a representation of the input sentence that captures the overall sentiment information embedded in it. The last hidden state of the BiLSTM i.e h_t, encapsulates the overall summary of the entire sentence's sentiment which we represent as cmu^c.

The forward state updates of the BiLSTM satisfy the following equations

$$\overrightarrow{f}_t = \sigma\left[\overrightarrow{W}_f\left[\overrightarrow{h}_{t-1}, \overrightarrow{r}_t\right] + \overrightarrow{b}_f\right] \tag{4.1}$$

$$\overrightarrow{i}_t = \sigma\left[\overrightarrow{W}_i\left[\overrightarrow{h}_{t-1}, \overrightarrow{r}_t\right] + \overrightarrow{b}_i\right] \tag{4.2}$$

$$\overrightarrow{o}_t = \sigma\left[\overrightarrow{W}_o\left[\overrightarrow{h}_{t-1}, \overrightarrow{r}_t\right] + \overrightarrow{b}_o\right] \tag{4.3}$$

$$\overrightarrow{l}_t = \tanh\left[\overrightarrow{V}\left[\overrightarrow{h}_{t-1}, \overrightarrow{r}_t\right] + \overrightarrow{d}\right] \tag{4.4}$$

$$\overrightarrow{c}_t = \overrightarrow{f}_t \cdot \overrightarrow{c}_{t-1} + \overrightarrow{i}_t \cdot \overrightarrow{l}_t \tag{4.5}$$

$$\overrightarrow{h}_t = \overrightarrow{o}_t \cdot \tanh(\overrightarrow{c}_t) \tag{4.6}$$

here σ is the logistic sigmoid function, $\overrightarrow{f}_t$, $\overrightarrow{i}_t$, $\overrightarrow{o}_t$ represent the forget, input and output gates respectively. $\overrightarrow{r}_t$ denotes the input at time t and $\overrightarrow{h}_t$ denotes the latent state, $\overrightarrow{b}_f$, $\overrightarrow{b}_i$, $\overrightarrow{b}_o$ and $\overrightarrow{d}$

represent the bias terms. The forget, input and output gates control the flow of information throughout the sequence. The backward states ($\overleftarrow{h}_1, \overleftarrow{h}_2, \ldots, \overleftarrow{h}_t$) are computed in a similar manner as above.

Finally, we represent the vector cmr^c as follows:

$$cmr^c = h^t = \begin{bmatrix} \overrightarrow{h}_t \\ \overleftarrow{h}_1 \end{bmatrix} \tag{4.7}$$

4.2.3.2 Specific Encoder

The graphical representation of this encoder can be seen from Fig. 4.2(B). The architecture of the Specific Encoder is very similar to the Collective Encoder, with one major difference: in this encoder, we employ a sub-word level attention mechanism. Our motive behind this is to be able to choose the sub-words which contribute the most towards the input's sentiment.

Different sub-words may encapsulate different information about sentiments, however it is crucial to identify the sub-words which play an important role in defining the overall sentiment of the sentence. The Specific Encoder helps in this by providing a context vector cmr^s.

In order to generate this context vector, we first concatenate the forward and backward states to obtain the annotations $(k_1, k_2, \ldots, k_t)$, where

$$k_i = \begin{bmatrix} \overrightarrow{k}_i \\ \overleftarrow{k}_i \end{bmatrix} \tag{4.8}$$

We then calculate the context vector cmr^s as follows:

$$cmr^s = \sum_{j=1}^{R} \alpha_j k_j \tag{4.9}$$

where α_i determines the part of the input sequence which should be emphasized or ignored and k_i stands for the output of the hidden units. α_i is computed by using a global context vector u_w as the query [63]. u_w is randomly initialized and learned during the training.

Using such a mechanism helps our model to adaptively select the more important sub-words from the less important ones.

4.2.3.3 Fusion of the Encoders

The fusion of the vectors obtained from both the encoders can be seen in Fig. 4.3. We concatenate the outputs obtained from both these encoders and use it as inputs to a fully connected neural network.

As shown in Fig. 4.2 and Fig. 4.3, we can see that h_t is incorporated into cmr^c to provide a summary of the sentiments present in the sentence. An important thing to notice is that, different encoding mechanisms will be evoked in both the encoders when trained jointly. The last hidden state of the collective encoder h_t plays a different role from that of k_t. The former has the responsibility to encode the sentiment present in the sequence of the sub-words which form the sentence, while the

latter is used for computing attention weights. Information obtained from both the encoders is utilized to come up with a unified representation of sentiment present in a sentence,

$$cmr^{sent} = [cmr^{c}; cmr^{s}] \tag{4.10}$$

where cmr^{sent} represents the unified representation of the sentiment.

4.2.4 Feature Network

In [40], researchers attempted sentiment analysis of tweets using classifiers trained on surface level features. We attempt to use similar linguistic features to augment the neural network framework of our model. These features are defined as:

- Capital words: Number of words which are in all capital letters
- Extended words: Number of words which have one or more contiguous repeating characters.
- Aggregate positive and negative sentiments: Using SentiWordNet [22] for every word except articles and conjunctions, and combining the sentiment polarity values into net positive aggregate and net negative aggregate features.
- Repeated exclamations and other punctuation: Number of sets of two or more contiguous exclamations, question marks, full stops, etc.
- Exclamation at end of sentence: Boolean value denoting whether the given sentence ends with an exclamation or not.
- Monolingual Sentence Vectors: While we do not have any suitable data to train word vectors for the code-mixed Hindi words in our input, we can reliably use the word representations for the English words in the input. We chose to use FastText sentence vectors [10] due to its support for learning vector representations of out-of-vocabulary words, which is useful for our dataset that contains Hindi words in Roman script.

4.2.5 Complete model - CMSA

As depicted in Fig. 4.2, the sub-word CNN feeds into a BiLSTM network with attention mechanism that models specific sentiments of the sentence, forming the *Specific Encoder*. The sub-word CNN is also connected to a BiLSTM in parallel that models the overall sentiment of the sentence, forming the *Collective Encoder*. The results from both these encoders are combined by concatenation, which is further combined with a Feature Network based on surface and semantic features, to augment the neural network learning.

A representation the complete model (CMSA) can be seen in Fig. 4.3. This architecture allows us to capture sentiment on the morpheme, syntactic and semantic levels simultaneously and learn specifically which parts of a sentence provide the most value to its sentiment classification.

With this system, we are able to combine the best of neural networks involving attention mechanisms with surface and semantic features that are used traditionally for sentiment analysis.

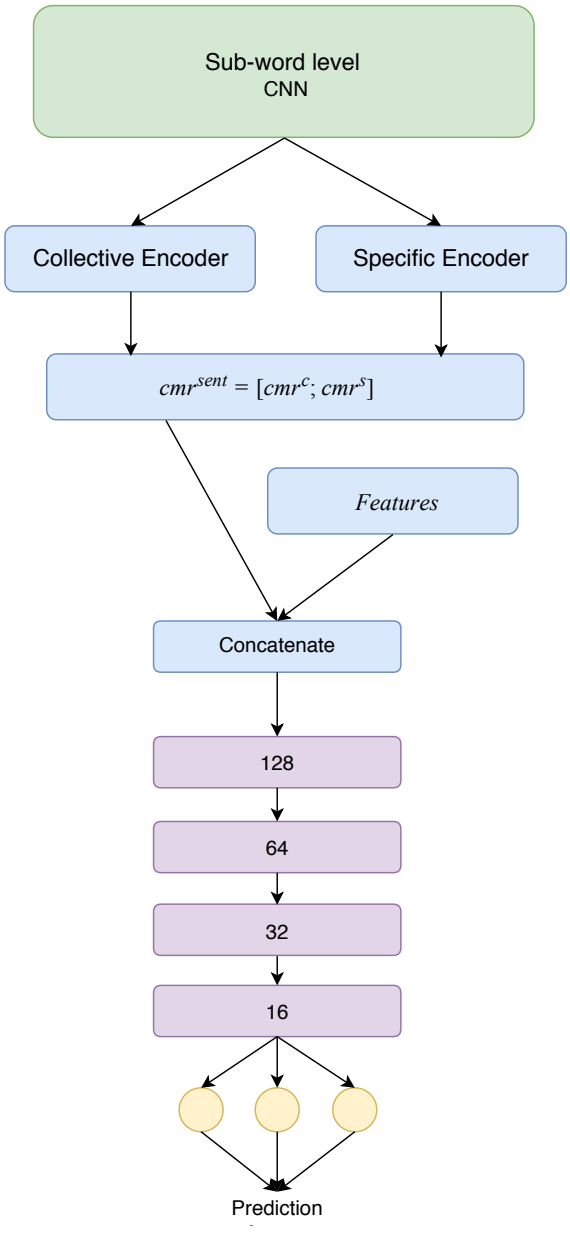

Figure 4.3: Complete Architecture of CMSA.

4.3 Experiments

In this section, we describe the dataset used, evaluation of our system with state-of-the-art and traditional approaches along with the parameters used for training the model.

4.3.1 Dataset

We make use of the dataset released by authors in [30], consisting of 3879 code-mixed English-Hindi sentences that they collected from public Facebook pages popular in India. The posts and comments on those pages attract comments from people across the country with varied sentiment polarity. This dataset contains 15% negative, 50% neutral and 35% positive sentences.

As is evident from examples in Table 4.1, there are some obvious problems in processing code-mixed data. The sentences are short, lack formally defined grammatical structure, and can contain spelling variations. Therefore, traditional approaches to sentiment analysis do not tend to work for code-mixed data.

Original Sentence	**English Translation**	**Sentiment Label**
I thought play *accha thha*	I thought that the play was good	**Positive**
Sorrrry , *aaj subah tak pata nhi tha* that I wudnt be able to come today :(	Sorry , Did not know until this morning that I wouldn't be able to come today :(	**Negative**
Trailer *dhannnsu hai bhai*	Trailer is amazing, brother	**Positive**

Table 4.1: Examples from the dataset

4.3.2 Baseline

We compare our approach with the methods proposed by the following:

- Wang and Manning, 2012 [60]: They use Naive Bayes (NB) and Multinomial Naive Bayes (MNB) methods alongside Support Vector Machines (SVM) and identify how using unigram and bigram models or their combination affects classification accuracy.

- Pang and Lee, 2008 [44]: The use an SVM based approach with ngram models to classify sentences based on their sentiments.

- Sharma et al., 2015 [56]: They perform word-level language identification of the code-mixed sentence, transliterate the English words from Roman to Devanagari and then perform sentiment analysis using a lexicon based approach.
- Joshi et al., 2016 [30]: Their system uses character embeddings of sentences as input to a Convolutional Neural Network that extracts sub-word level information from the sentence. LSTMs are then used to propagate relevant information.
- Choudhary et al., 2018 [16]: They introduced a framework called *Sentiment Analysis of Code-Mixed Text (SACMT)*, which utilizes siamese networks to map code-mixed and standard sentences to a common sentiment space and then classifies sentiment using contrastive learning.
- Joulin et al., 2016 [31]: Their system, FastText, performs classification tasks by incorporating sub-word information in word vectors and is able to handle out-of-vocabulary words as well. They provide an API for using their system along with pre-trained models for vector representations.

4.3.3 Parameter Learning

We use an Intel(R) Xeon(R) CPU E5-2658 v3 @ 2.20GHz with 128GB RAM and GeForce GTX 1080 GPU. The system architecture has been implemented in Keras [15]. The dataset is randomly divided into training and validation sets in a 4:1 ratio. The hyperparameters of our model are trained using the validation set. The proposed model and all variants described in this paper are learned by optimizing the validation accuracy. We used a batch size of 128 and used AdaMax [33] as the optimizer. We use categorical crossentropy for learning the parameters of the model.

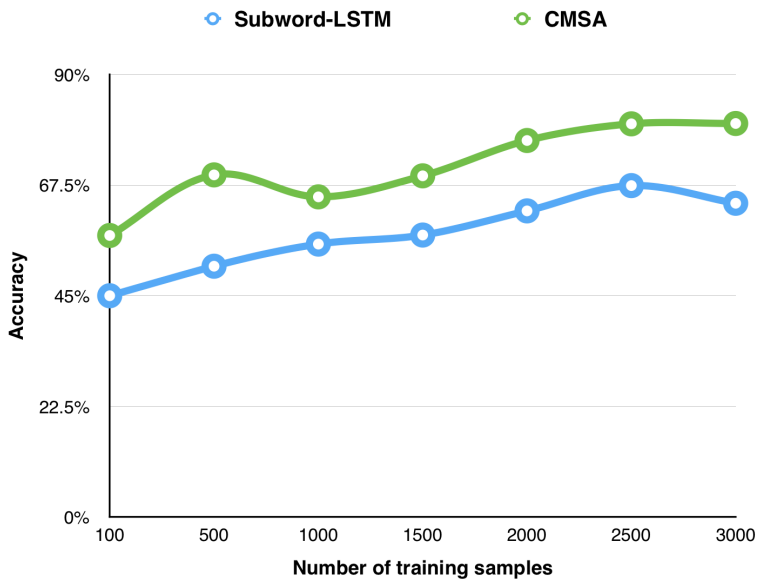

Figure 4.4: Performance comparison with baseline for varying number of training sentences

4.4 Results and Analysis

In this section, we present the results obtained by various experiments that were run with our system.

Method	Reported in	Accuracy	F1-score
SVM (Unigram)	Pang and Lee, 2008 [44]	57.6%	0.5232
SVM (Uni+Bigram)	Pang and Lee, 2008 [44]	52.96%	0.3773
NBSVM (Unigram)	Wang and Manning, 2012 [60]	59.15%	0.5335
NBSVM (Uni+Bigram)	Wang and Manning, 2012 [60]	62.5%	0.5375
MNB (Unigram)	Wang and Manning, 2012 [60]	66.75%	0.6143
MNB (Uni+Bigram)	Wang and Manning, 2012 [60]	66.36%	0.6046
MNB (Tf-Idf)	Wang and Manning, 2012 [60]	63.53%	0.4783
Lexicon Lookup	Sharma et al., 2015 [56]	51.15%	0.252
Char-LSTM	Joshi et al., 2016 [30]	59.8%	0.511
Subword-LSTM	Joshi et al., 2016 [30]	69.7%	0.658
FastText	Joulin et al., 2017 [31]	46.39%	0.505
SACMT	Choudhary et al., 2018 [16]	77.3%	0.759
CMSA	Proposed	**83.54%**	**0.827**

Table 4.2: Results show that proposed system performs significantly better over state-of-the-art methods for code-mixed Sentiment Analysis

4.4.1 Performance comparison with baselines

We compared our method with the baselines. From Table 4.2, it can be clearly observed that CMSA outperforms the state-of-the-art techniques by 6.24% in accuracy and 0.068 in F1-score.

We can observe a trend in the performances among the baselines themselves. The lexicon lookup approach performs the worst on our dataset among all the methods, likely because of the high number of spelling variations or misspellings in our data that it is unable to handle correctly. Naive Bayes methods perform fairly well, with Multinomial Naive Bayes performing better than simple SVM or SVM in combination with Naive Bayes.

The methods Char-LSTM, Subword-LSTM and SACMT all employ deep learning techniques to perform the task of sentiment analysis, and it can be observed that they perform better than traditional approaches by a fair margin. The significant difference in accuracy between Subword-LSTM

and Char-LSTM, as seen in table 4.2, confirms our assumptions from section 4.2.2 about subword-level representations being better for this use case as compared to character-level embeddings.

However, deep learning techniques on their own do not suffice for the task. In order to augment these techniques, we involve the use of specific linguistic features, based on real world knowledge of code-mixing. Ultimately, this helps our system learn classification of sentiments better than the above described methods even on fewer training samples, as can be observed from Figure 4.4 where we compare against Subword-LSTM model which is architecturally most similar to our model but does not use linguistic features or attention.

4.4.2 Effect of different Encoders

We note the effects of varying the kind of recurrent network used. We compare the results obtained by using BiLSTMs, LSTMs, GRU and Vanilla RNN. From Table 4.3, the observable trend in performance is: BiLSTM >LSTM >GRU >RNN, although the differences are not substantial. One of the reasons for this could be that a LSTM or a GRU can handle long term dependencies better than an RNN, and a BiLSTM works better than an LSTM particularly when sentiment information may be embedded towards the beginning of the sentence.

We also experimented with variants of our own model, by replacing the fused encoders with the Specific Encoder and the Collective Encoder separately. From Table 4.5, we can see that the trend in performance is as follows: CMSA >Collective Encoder >Specific Encoder. This shows that merely learning the overall sentiment embedded in a sentence is not enough, and neither is just learning the sentiments associated with the sub-words themselves. A combination of the two encoders provides a better sentiment classification model.

4.4.3 Effect of using Feature Network

One of the most distinguishing aspects of our work from previous approaches for this task involves the augmentation of deep learning techniques with a Feature Network. We experimented by performing sentiment analysis using just the deep learning framework, just the feature network and finally, the combination of the two. We report the results in Table 4.6, where it can be easily observed that the Feature Network is able to positively augment the classification by the neural network.

4.4.4 Effect of varying the number of hidden layers

We observe the performance of our system while varying the number of hidden layers in the neural network. We experiment with using one layer of size 128, two layers of sizes 128 and 64 each, three layers of sizes 128, 64, 32 each and four layers of sizes 128, 64, 32, 16 each. We find that the best result is obtained in the case of four layers described above.

Encoder	Accuracy	F1-score
RNN	75.91%	0.699
GRU	78.58%	0.768
LSTM	78.5%	0.758
BiLSTM (CMSA)	**83.54%**	**0.827**

Table 4.3: Performance using different encoders

Layout of layers	Accuracy	F1-score
128	77.67%	0.766
128 - 64	74.87%	0.717
128 - 64 - 32	73.03%	0.685
128 - 64 - 32 - 16	**83.54%**	**0.827**

Table 4.4: Effect of varying number of hidden layers

Method	Accuracy	F1-score
Specific Encoder	80.2%	0.801
Collective Encoder	77.3%	0.795
Specific + Collective (CMSA)	**83.54%**	**0.827**

Table 4.5: Performance using different encoding mechanisms with Feature Network

System	Accuracy	F1-score
Encoders only	75.74%	0.705
Feature Network only	57.9%	0.381
CMSA	**83.54%**	**0.827**

Table 4.6: Performance using Feature Network and Encoders separately and combined

4.5 Conclusion

We propose a neural network architecture for sentiment analysis of English-Hindi code-mixed data that improves on traditional and state-of-the-art approaches. We use a hybrid approach that combines recurrent neural networks utilizing attention mechanisms, with surface features, yielding a unified representation that can be trained to classify sentiments. We conducted extensive experiments on a real world dataset consisting of online social media text, and demonstrated that our system is able to achieve a sentiment classification accuracy of 83.54% and an F1-score of 0.827, outperforming state-of-the-art approaches for this task.

Chapter 5

Other applications of our Hybrid Feature-Attention model

The Hybrid Feature-Attention model, which we have described in detail in Chapter 4, was developed specifically for the task of Sentiment Analysis of Code-Mixed data. We name it so due to its hybrid architecture of BiLSTMs with attention mechanisms combined with a Feature Network based on surface and semantic features that augments the neural network learning. However, we found that we are able to achieve good results on application of the same model to other tasks such as Clickbait Detection. By including surface and semantic level features in the training of the neural network for these tasks, we were able to achieve a high degree of accuracy.

We now discuss the application of our model to clickbait detection in Natural Language Processing and describe the experiments performed.

5.1 Clickbait Detection

In recent years, the preferred medium for the delivery of content has shifted from magazines and other offline sources to the Internet. Most people use the Internet as their source of information as it is easier and faster than conventional methods.

In an effort to keep up with the changing times and increase their reach, media outlets have systematically grown their online presence over the years. Their primary sources of revenue are advertisements on their websites, apart from subscription services for their content. With so much information available to everyone, the user has his pick in choosing the content they want. This results in a competition between the media outlets for the user's attention as more attention leads to higher advertisement based revenue. However, the craze of generating more and more ad-based revenue by these media outlets has come at a cost to the readers. These outlets are now using techniques called *clickbait* to boost click-through rates of their adverts.

Clickbait can be a headline or other attention grabbing snippet of an article's description that encourages readers to click on hyperlinks leading to the article, especially when the value of the article's content is dubious, as defined by Merriam Webster. Clickbait creates, and capitalizes, on the Loewen-

stein gap [38] by increasing expectations from a story on the Internet on purpose, through headlines, attached images or related text.

Using a similar hybrid architecture described in Chapter 5, we use a two-pronged approach to detecting headlines which could act as clickbait. The first component uses the distributional semantics of the article's textual content by modelling its sequential properties. The article's headline is represented using its sub-word level embeddings. This sub-word level representation serves as input to a Bi-directional Long Short Term Memory (Bi-LSTM) network, which is connected to an attention layer and finally through a dense layer. The second component uses Doc2Vec embeddings of the headline and article content. It performs element-wise multiplication for both of them and its output is concatenated with the output of the first component. The final result then goes through multiple hidden layers to yield the classification.

Previous work related to this task has used either word-level or character-level representations. Instead, we propose the use of sub-word level representations since it also encapsulates the morphological features of the words. Using an attention mechanism helps the model identify the surprise corresponding to each representation of the headline and article text.

5.2 Model Architecture

We now describe our approach to clickbait detection and the reasons behind devising such a model. Our approach is presented as a fusion of two components, each exploiting a particular type of embedding: (1) BiLSTM with attention, and (2) Doc2Vec enrichment. Figure 5.2 lays out our proposed architecture.

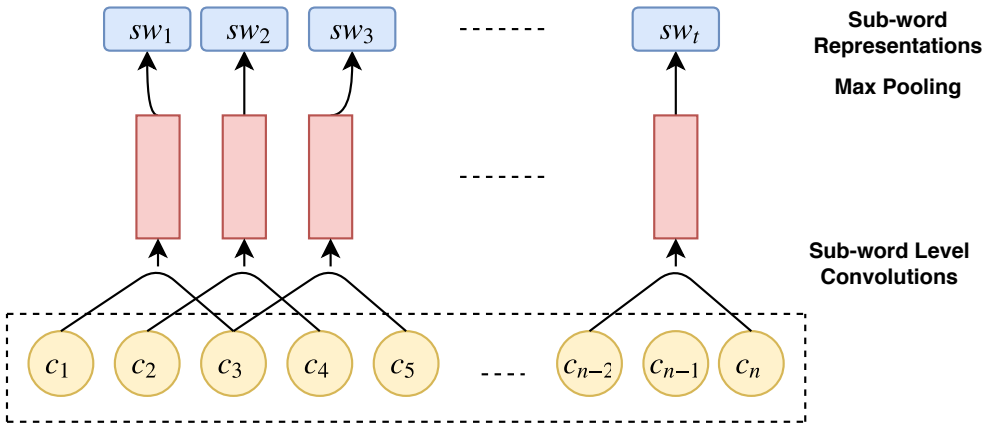

Figure 5.1: Architecture for learning Sub-word Level Representations using CNN

We start with an explanation of the various types of embeddings we have used and proceed to describe the various components of our model, both individually and together. Finally, we cover how the parameters are learned.

5.2.1 Sub-word Level Representation

Word2Vec [39] has fast become the most popular text embedding method for text since it models a word based on its context. In [32], researchers proposed a convolutional neural network architecture to generate subword-level representations of words in order to capture word orthography. Sub-word level embeddings learn representations for character n-grams and represent words as the sum of the n-gram vectors [9]. Such representations also take into account word roots and inflections, rather than just word context. They work well even with highly noisy text with containing misspellings due to the model learning morpheme-level feature maps. They have proven to be extremely useful in tasks such as sentiment analysis [49], PoS tagging [46] and language modeling [32]. These intermediate sub-word feature representations are learned by the filters during the convolution operation. We generate such an embedding by passing the characters of a sentence individually into 3 layer 1D convolutional neural network. Each filter then acts as a learned sub-word level feature. A representation for this architecture can be found in Figure 5.1.

5.2.2 Document Embeddings

Doc2Vec [37] is an unsupervised approach to generate vector representations for slightly larger bodies of text, such as sentences, paragraphs and documents. It has been adapted from Word2Vec [39] which is used to generate vectors for words in large unlabeled corpora. The vectors generated by this approach come handy in tasks like calculating similarity metrics for sentences, paragraphs and documents. In sequential models like RNNs, the word sequence is captured in the generated sentence vectors. However, in Doc2Vec, the representations are order independent. We use GenSim [52] to learn 300 dimensional Doc2Vec embeddings for each target description and post title available.

5.2.3 Bidirectional LSTM with Attention

Recurrent Neural Network (RNN) is a class of artificial neural networks which utilizes sequential information and maintains history through its intermediate layers. A standard RNN has an internal state whose output at every time-step which can be expressed in terms of that of previous time-steps. However, it has been seen that standard RNNs suffer from a problem of vanishing gradients [28]. This means it will not be able to efficiently model dependencies and interactions between sub-word representations that are a few steps apart. LSTMs are able to tackle this issue by their use of gating mechanisms. We convert each article headline into its corresponding sub-word level representation to act as input to our bidirectional LSTMs.

$(\overrightarrow{h}_1, \overrightarrow{h}_2, \ldots, \overrightarrow{h}_R)$ represent forward states of the LSTM and its state updates satisfy the following equations:

$$[\overrightarrow{f_t}, \overrightarrow{i_t}, \overrightarrow{o_t}] = \sigma[\overrightarrow{W}[\overrightarrow{h}_{t-1}, \overrightarrow{r_t}] + \overrightarrow{b}] \tag{5.1}$$

$$\overrightarrow{l_t} = \tanh[\overrightarrow{V}[\overrightarrow{h}_{t-1}, \overrightarrow{r_t}] + \overrightarrow{d}] \tag{5.2}$$

$$\overrightarrow{c_t} = \overrightarrow{f_t} \cdot \overrightarrow{c}_{t-1} + \overrightarrow{i_t} \cdot \overrightarrow{l_t} \tag{5.3}$$

$$\overrightarrow{h_t} = \overrightarrow{o_t} \cdot \tanh(\overrightarrow{c_t}) \tag{5.4}$$

here σ is the logistic sigmoid function, $\overrightarrow{f_t}$, $\overrightarrow{i_t}$, $\overrightarrow{o_t}$ represent the forget, input and output gates respectively. $\overrightarrow{r_t}$ denotes the input at time t and $\overrightarrow{h_t}$ denotes the latent state, $\overrightarrow{b_t}$ and $\overrightarrow{d_t}$ represent the bias terms. The forget, input and output gates control the flow of information throughout the sequence. $\overrightarrow{W}$ and $\overrightarrow{V}$ are matrices which represent the weights associated with the connections.

$(\overleftarrow{h}_1, \overleftarrow{h}_2, \ldots, \overleftarrow{h}_R)$ denote the backward states and its updates can be computed similarly.

The number of bidirectional LSTM units is set to a constant *K*, which is the maximum length of all title lengths of records used in training. The forward and backward states are then concatenated to obtain $(h_1, h_2, \ldots, h_K)$, where

$$h_i = \begin{bmatrix} \overrightarrow{h}_i \\ \overleftarrow{h}_i \end{bmatrix} \tag{5.5}$$

Finally, we are left with the task of figuring out the significance of each word in the sequence i.e. how much a particular sub-word representation influences the clickbait-y nature of the post. The effectiveness of attention mechanisms have been proven for the task of neural machine translation [4] and it has the same effect in this case. The goal of attention mechanisms in such tasks is to derive context vectors which capture relevant source side information and help predict the current target representation. The sequence of annotations generated by the encoder to come up with a context vector capturing how each sub-word contributes to the record's clickbait quotient is of paramount importance to this model. In a typical RNN encoder-decoder framework [4], a context vector is generated at each time-step to predict the target sub-word. However, we only need it for calculation of context vector for a single time-step.

$$c_{attention} = \sum_{j=1}^{K} \alpha_j h_j \tag{5.6}$$

where, h_1,...,h_K represents the sequence of annotations to which the encoder maps the post title vector and each α_j represents the respective weight corresponding to each annotation h_j. This is represented as the left most component in Figure 5.2.

5.2.4 Doc2Vec Enrichment

Each record in the dataset has a target description attached with it. This is the entire text of the article whose title has been given. By definition, clickbait articles differ from the content described in their headline. We generate document embeddings for both the title and the article text and perform

element wise multiplication over the two. This allows us to capture the interaction between the two, something which has not been used before. Since the title is supposed to mislead the reader with respect to the content, modeling this interaction in terms of their similarity gives an added dimenstion to our approach. It augments the output obtained from the first component.

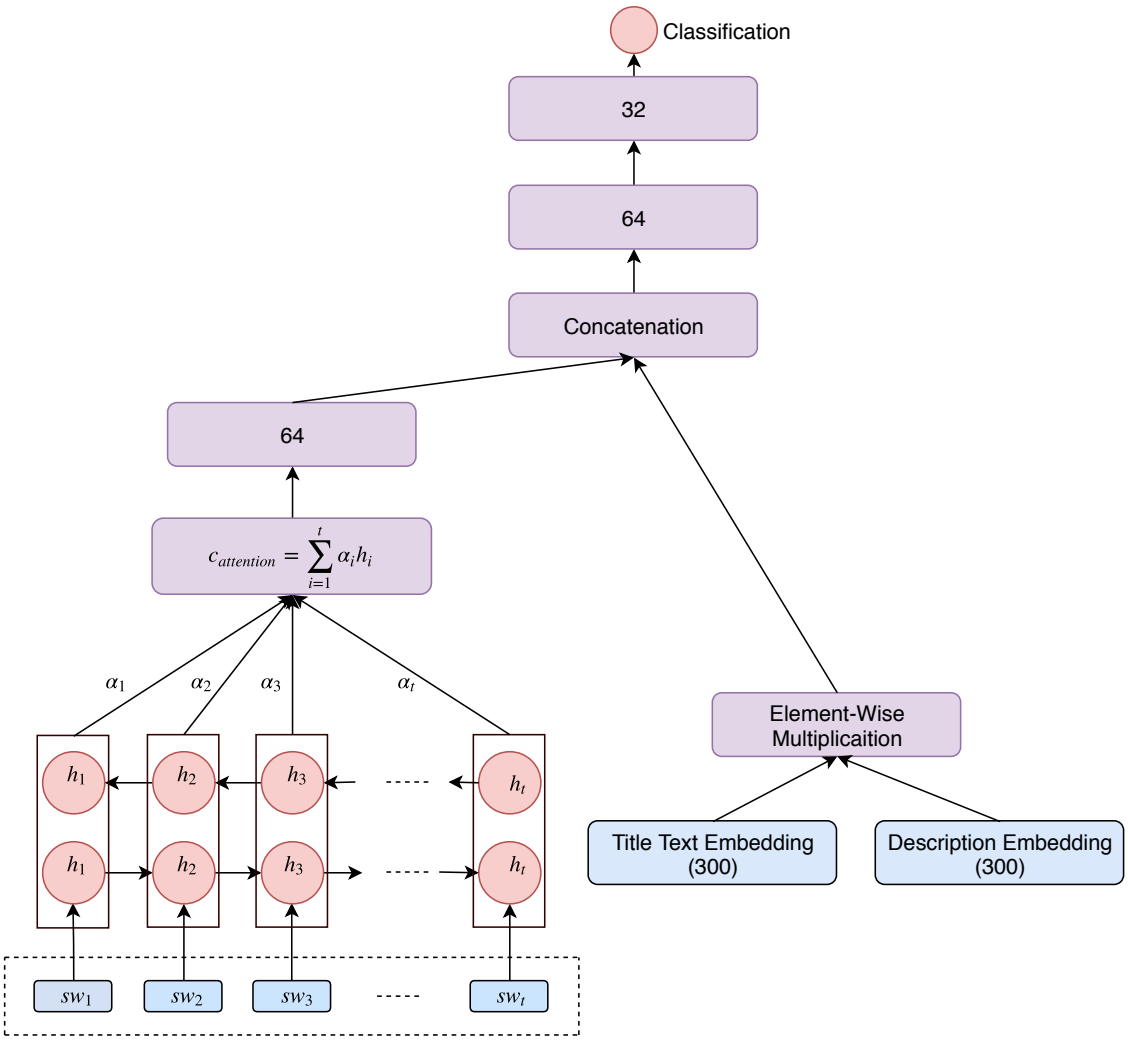

Figure 5.2: Complete Model Architecture

5.2.5 Fusion of Components

The outputs from the aforementioned components are now concatenated and passed through two dense layers and finally goes into a fully connected layer. This layer finally gives out the probability that a post can be marked clickbait.

5.2.6 Learning the Parameters

We use binary cross-entropy as the loss optimization function for our model. The cross-entropy method [21] is an iterative procedure where each iteration can be divided into two stages:

(1) Generate a random data sample (vectors, trajectories etc.) according to a specified mechanism.

(2) Update the parameters of the random mechanism based on the data to produce a "better" sample in the next iteration.

5.3 Evaluation and Results

In [47], researchers crowdsourced the annotation of 19538 tweets they had curated, into various levels of their clickbait-y nature. These tweets contained the title and text of the article and also included supplementary information such as target description, target keywords and linked images. We trained our model over 17000 records in the described dataset and test it over 2538 disjoint instances from the same. We performed our experiments with the aim of increasing the accuracy and F1 score of the model. Other metrics like mean squared error (MSE) were also considered.

5.3.1 Training

We randomly partition the training set of over 17000 posts into training and validation set in a 4:1 ratio. This ensures that the two sets do not overlap. The model hyperparameters are tuned over the validation set. We initialise the fully connected network weights with the uniform distribution in the range $-\sqrt{6/(fanin + fanout)}$ and $\sqrt{6/(fanin + fanout)}$ [24]. We used a batch size of 256 and adadelta [64] as a gradient based optimizer for learning the model parameters.

Model	F1 Score	Accuracy
Proposed Approach	**0.63**	**83.49**%
BiLSTM [2]	0.61	83.28%
Feature Engineering SotA [48]	0.55	83.24%
Concatenated NN Architecture [58]	0.39	74%

Table 5.1: Model Performance Comparison

5.3.2 Model Comparison

In Table 1, we evaluate our model against the existing state-of-the-art for the dataset used and other models which have employed similar techniques to accomplish the task. It is clear that our proposed model outperforms the previous feature engineering benchmark and other work done in the field both

in terms of F1 score and accuracy of detection. Feature engineering models rely on a selection of handcrafted attributes which may not be able to consider all the factors involved in making a post clickbait. The approach proposed by the authors in [58] takes into account each of the textual features available in an individual fashion, considering them to be independent of each other, which is not the case since, by definition of clickbait, the content of the article title and text are not mutually exclusive. The authors in [35] proposed the integration of multimodal embeddings. In [2], researchers utilise word and character embeddings which do not capture morpheme-level information that may incorporate a surprise element.

5.4 Summary

We have developed an approach for the task of clickbait detection that learns morphological features through sub-word representations. These embeddings are modeled by the LSTM network with Attention mechanism, which allows it to learn the nature of the article from individual representations. Document embeddings for headline and article content augment the generated embeddings and are finally used to classify the article. In the future, it would be interesting to explore if integrating the sub-word representations with deep learning models the temporal and sequential properties of text better.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we present a study of code-mixing in natural language on online social media. We develop systems that facilitate the tasks of machine translation and sentiment analysis of such data.

To promote the task of machine translation of code-mixed data, we have created a set of 6096 English-Hindi code-mixed and monolingual English gold standard parallel sentences. We hope that this can motivate the generation of larger data resources for this domain.

As machine translation systems require a large number of parallel sentences, it is not feasible to develop specialized MT systems for code-mixed data currently. In order to improve code-mixed translations by existing MT systems, we presented a pre-processing pipeline, that can be used to augment existing MT systems such that translation of code-mixed data can be improved without training an MT system specifically for code-mixed text. We compared the performance of existing phrase based and neural network based MT systems on English-Hindi code-mixed data, with and without the use of our augmentation pipeline. Using the evaluation metrics selected, we show that there is a quantifiable improvement in the accuracy of translations with the augmentation proposed. As part of our study, we have observed that long distance re-ordering of words is still an issue with code-mixed MT. Also, since code-mixed language does not have a standard form, it is difficult to establish a correct version of spelling for a particular word. Language identification is the most critical module for translation augmentation, because the same orthographic form can lead to valid words in multiple languages.

We also proposed a neural network architecture for sentiment analysis of English-Hindi code-mixed data that improves on traditional and state-of-the-art approaches. We develop a hybrid approach that combines recurrent neural networks utilizing attention mechanisms, with surface features, yielding a unified representation that can be trained to classify sentiments, which we call the "Feature-Attention" model. We conducted extensive experiments on a real world dataset consisting of online social media text, and demonstrated that our system is able to achieve a sentiment classification accuracy of 83.54% and an F1-score of 0.827, outperforming state-of-the-art approaches for this task.

Finally, we applied a model based on our hybrid neural network architecture infused with surface features, to the task of clickbait detection in online articles, and we were able to report that our model outperforms the previous feature engineering benchmark and other approaches by scoring an accuracy of 83.49% and an F1-score of 0.63.

6.2 Future Work

The work presented in this thesis also motivated us to search for other avenues of research in the domain of Code-mixing and the processing of natural language content generated by users on the Internet. Building upon this study, one can explore new techniques in various tasks processing code-mixed data like question answering systems, discourse analysis, text summarization, etc.

Creation of larger data resources would be a good first step in moving this research forward. Research based on deep learning, which works well for other domains, is quite restricted due to lack of abundant code-mixed data resources. We hope that the creation and release of the English-Hindi code-mixed to monolingual English parallel corpus during the course of our study will motivate further work in the development of data resources for code-mixing, and we aim to generate more of such datasets to facilitate research in this domain.

Another important aspect is adding support for more languages in these systems. Our work has focused predominantly on the code-mixed language formed by English and Hindi. However, there is a lot of work that can be done for various other Indian languages such as Bengali, Gujarati and Telugu, which are also widely code-mixed with English and Hindi.

Taking the work of our current study further, there are other directions that can be explored for the development of tools for code-mixed social media text. One of the possibilities is the creation of a complete end-to-end machine translation system designed specifically with code-mixed data in mind, which will become more viable as more data resources for this task are available. Another avenue to explore is the use of our Hybrid-Attention model for the task of Aggression detection on social media websites, as it can also be modelled as a classification task.

Finally, there are improvements that can be made to the way feature engineering is performed at surface and semantic levels such that the Feature Network component of our model architecture can be made more accurate. With the use of better and more accurate normalization systems for code-mixed data, it might be possible to normalize non-standard variations of words in the code-mixed sentence such that vector representations of such words become reasonably accurate to use in such systems.

Related Publications

1. Mrinal Dhar and Vaibhav Kumar and Manish Shrivastava, 2018, June. **Enabling Code-Mixed Translation: Parallel Corpus Creation and MT Augmentation Approach**. In Linguistic Resources for NLP Workshop, 27th International Conference on Computational Linguistics (LR4NLP, COLING 2018)

2. Vaibhav Kumar, Mrinal Dhar, Dhruv Khattar, Yash Kumar Lal, Abhimanshu Mishra, and Manish Shrivastava, Vasudeva Varma. 2018, June. **SWDE : A Sub- Word And Document Embedding Based Engine for Clickbait Detection**. In Proceedings of SIGIR 2017 Workshop on Computational Surprise in Information Retrieval (CompS18, SIGIR).

3. Mrinal Dhar and Vaibhav Kumar and Manish Shrivastava, 2018. **Looking Beyond the Obvious: Code-Mixed Sentiment Analysis (CMSA)**. Submitted to 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)

Bibliography

[1] Nlp tools contest @ icon 2017 : 14th international conference on natural language processing, 2017.

[2] A. Anand, T. Chakraborty, and N. Park. We used Neural Networks to Detect Clickbaits: You won't believe what happened Next! In *Advances in Information Retrieval. 39th European Conference on IR Research (ECIR 17)*, Lecture Notes in Computer Science. Springer, 2017.

[3] O. Araque, I. Corcuera-Platas, J. F. Snchez-Rada, and C. A. Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77:236 – 246, 2017.

[4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] K. Bali, J. Sharma, M. Choudhury, and Y. Vyas. "i am borrowing ya mixing?" an analysis of english-hindi code mixing in facebook. In *In Proceedings of the First Workshop on Computational Approaches to Code Switching, EMNLP. Monica Stella Cardenas-Claros and Neny Isharyanti*, 2014.

[6] U. Barman, A. Das, J. Wagner, and J. Foster. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of The First Workshop on Computational Approaches to Code Switching*, pages 13–23, 2014.

[7] I. A. Bhat, V. Mujadia, A. Tammewar, R. A. Bhat, and M. Shrivastava. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14, pages 48–53, New York, NY, USA, 2015. ACM.

[8] P. Biyani, K. Tsioutsiouliklis, and J. Blackmer. "8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 2016.

[9] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[10] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[11] J. Carrera, O. Beregovaya, and A. Yanishevsky. Machine translation for cross-language social media, 2009.

[12] A. Chakraborty, B. Paranjape, S. Kakarla, and N. Ganguly. Stop clickbait: Detecting and preventing clickbaits in online news media. *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 9–16, 2016.

[13] K. R. Chandu, M. Chinnakotla, A. W. Black, and M. Shrivastava. Webshodh: A code mixed factoid question answering system for web. In G. J. Jones, S. Lawless, J. Gonzalo, L. Kelly, L. Goeuriot, T. Mandl, L. Cappellato, and N. Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 104–111, Cham, 2017. Springer International Publishing.

[14] G. Chittaranjan, Y. Vyas, K. Bali, and M. Choudhury. Word-level language identification using crf: Code-switching shared task report of msr india system. page 7379, Doha, Qatar, October 2014. Association for Computational Linguistics.

[15] F. Chollet et al. Keras. `https://keras.io`, 2015.

[16] N. Choudhary, R. Singh, I. Bindlish, and M. Shrivastava. Sentiment analysis of code-mixed languages leveraging resource rich languages. 04 2018.

[17] M. Choudhury, R. Saraf, V. Jain, A. Mukherjee, S. Sarkar, and A. Basu. Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(3):157–174, Dec 2007.

[18] D. Crystal et al. *Internet linguistics: A student guide*. Routledge, 2011.

[19] B. Danet and S. C. Herring. *The multilingual Internet: Language, culture, and communication online*. Oxford University Press on Demand, 2007.

[20] A. Das and B. Gambäck. Identifying languages at the word level in code-mixed indian social media text. 2014.

[21] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, Feb 2005.

[22] A. Esuli and F. Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06*, pages 417–422, 2006.

[23] M. Giatsoglou, M. G. Vozalis, K. Diamantaras, A. Vakali, G. Sarigiannidis, and K. C. Chatzisavvas. Sentiment analysis leveraging emotions and word embeddings. *Expert Systems with Applications*, 69:214 – 224, 2017.

[24] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[25] J. J. Gumperz. *Discourse strategies*, volume 1. Cambridge University Press, 1982.

[26] S. Gupta, P. Bansal, and R. Mamidi. Resource creation for hindi-english code mixed social media text. 07 2016.

[27] S. C. Herring. Media and language change: Introduction. *Journal of Historical Pragmatics*, 4(1):1–17, 2003.

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

[29] A. Jamatia, B. Gambäck, and A. Das. Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. Association for Computational Linguistics, 2015.

[30] A. Joshi, A. Prabhu, M. Shrivastava, and V. Varma. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491. The COLING 2016 Organizing Committee, 2016.

[31] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.

[32] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 2016.

[33] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[34] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[35] V. Kumar, D. Khattar, Y. K. Lal, and V. Varma. Identifying clickbait: A multi-strategy approach using neural networks. In *Proceedings of the 41sr International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18, 2018.

[36] A. Kunchukuttan, P. Mehta, and P. Bhattacharyya. The IIT bombay english-hindi parallel corpus. *CoRR*, abs/1710.02855, 2017.

[37] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[38] G. Loewenstein. The psychology of curiosity: A review and reinterpretation. 116:75–98, 07 1994.

[39] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[40] S. M. Mohammad, S. Kiritchenko, and X. Zhu. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *CoRR*, abs/1308.6242, 2013.

[41] C. Myers-Scotton. Common and uncommon ground: Social and structural factors in codeswitching. *Language in society*, 22(4):475–503, 1993.

[42] C. Myers-Scotton. *Duelling Languages: Grammatical Structure in Codeswitching*. Clarendon Press, 1997.

[43] D. Nguyen and A. S. Doğruöz. Word level language identification in online multilingual communication. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 857–862, 2013.

[44] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, Jan. 2008.

[45] B. G. Patra, D. Das, and A. Das. Sentiment analysis of code-mixed indian languages: An overview of sail_code-mixed shared task @icon-2017. *CoRR*, abs/1803.06745, 2018.

[46] B. Plank, A. Søgaard, and Y. Goldberg. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418. Association for Computational Linguistics, 2016.

[47] M. Potthast, T. Gollub, K. Komlossy, S. Schuster, M. Wiegmann, E. Garces, M. Hagen, and B. Stein. Crowdsourcing a Large Corpus of Clickbait on Twitter. In *(to appear)*, 2017.

[48] M. Potthast, S. Köpsel, B. Stein, and M. Hagen. Clickbait Detection. In N. Ferro, F. Crestani, M.-F. Moens, J. Mothe, F. Silvestri, G. Di Nunzio, C. Hauff, and G. Silvello, editors, *Advances in Information Retrieval. 38th European Conference on IR Research (ECIR 16)*, volume 9626 of *Lecture Notes in Computer Science*, pages 810–817, Berlin Heidelberg New York, Mar. 2016. Springer.

[49] A. Prabhu, A. Joshi, M. Shrivastava, and V. Varma. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. *CoRR*, abs/1611.00472, 2016.

[50] A. Pravalika, V. Oza, N. P. Meghana, and S. S. Kamath. Domain-specific sentiment analysis approaches for code-mixed social network data. In *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, volume 00, pages 1–6, July 2017.

[51] K. C. Raghavi, M. K. Chinnakotla, and M. Shrivastava. "answer ka type kya he?": Learning to classify questions in code-mixed language. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 853–858, New York, NY, USA, 2015. ACM.

[52] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, 2010.

[53] S. Rijhwani, R. Sequiera, M. C. Choudhury, and K. Bali. Translating code-mixed tweets: A language detection based system. In *3rd Workshop on Indian Language Data Resource and Evaluation - WILDRE-3*, 2016.

[54] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[55] A. Sharma, S. Gupta, R. Motlani, P. Bansal, M. Srivastava, R. Mamidi, and D. M. Sharma. Shallow Parsing Pipeline for Hindi-English Code-Mixed Social Media Text. *ArXiv e-prints*, Apr. 2016.

[56] S. Sharma, P. Srinivas, and R. C. Balabantaray. Text normalization of code mix and sentiment analysis. *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1468–1473, 2015.

[57] R. M. K. Sinha and A. Thakur. Machine translation of bi-lingual hindi-english (hinglish) text. *10th Machine Translation summit (MT Summit X), Phuket, Thailand*, pages 149–156, 2005.

[58] P. Thomas. Clickbait identification using neural networks. *CoRR*, abs/1710.08721, 2017.

[59] Y. Vyas, S. Gella, J. Sharma, K. Bali, and M. Choudhury. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 974979, Doha, Qatar, October 2014. Association for Computational Linguistics.

[60] S. Wang and C. D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[61] M. Warschauer, G. R. E. Said, and A. G. Zohry. Language choice online: Globalization and identity in egypt. *Journal of Computer-Mediated Communication*, 7(4):JCMC744, 2002.

[62] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, ukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[63] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, 2016.

[64] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[65] Y. Zhou. Clickbait detection in tweets using self-attentive network. *CoRR*, abs/1710.05364, 2017.