Sarcasm Detection System for Hinglish Language (SDSHL)

A dissertation is presented towards the

fulfilment of the requirements for M.Sc. in Data Science

Hari Thapliyal

Student Number: PN927682

Faculty Advisor: Dr. Anil Vuppala

IIIT Hyderabad

Dissertation Submitted to

Liverpool John Moore University, UK

Date:05-Nov-2020

DEDICATION

First, I dedicate this work to those millions of Hinglish speaking people of the world who gave rise to this phenomenon, which we call English language. Their verbal and written communication style makes NLP & NLU work more complex for the machines.

Second, I dedicate this work to those NLP, NLU experts who are working day and night to improve algorithms and make systems so that the mental distance between those who do not speak the same language can be removed or minimized at least.

Third, I dedicate this work to those smart people who, when annoyed, upset, or frustrated they chose words carefully and craft message in a positive way so that message goes straight to the heart of the receiver. We call that message a sarcastic message.

Without these three communities around me I could have never thought of doing this project.

ACKNOWLEDGEMENTS

First, I would like to thank Prof Anil Vuppala of IIIT Hyderabad, who have been a positive and decisive force behind this work. Out of many topics which I listed for my project; Prof Anil told me the value of this project. Since the day this topic is selected to writing last word of this dissertation, he always remained a guiding factor and connecting me to right people and resources around him. He is such soft spoken and positive person that I will never forget the value of time which he spent to provide the best possible guidance from his side.

Second, I want to thank Mr. Rajiv Saxena. He has been Chief Editor of Rastriya Sahara Group. One of the leading daily Hindi newspaper in India. Being native speaker and SME, his understanding of the nuisance of Hinglish is commendable. Because of his busy schedule, it was not easy to engage him in this project but out of love for me and the value of this work he spent time in doing some important discussion with me. He also spent his valuable time in the sentence annotation work. I remember long discussion with him where we were debating a particular sentence is sarcasm or not.

Third, I want to thank Shruti Tripati. She is wife of my close friend and like my younger sister. When I was looking for an annotator on my project. She volunteered for this work willingly and completed her part of the work before time.

ABSTRACT

Hinglish is third[1] most spoken language on the planet. (Wikipage, 2020a) states that 65% of Indian population is under 35 years age. Several disruptions like low cost mobile phone, extremely cheap data, digital India initiatives by government of India has caused huge surge in Hinglish language content. Hinglish language context is available in audio, video, images, and text format. We can find Hinglish content in comment box of product, new articles, service feedback, WhatsApp, social media like YouTube, Facebook, twitter etc. To engage with consumer, it is extremely important to analyse the sentiments, but to perform sentiment analysis it is not possible to read every comment or feedback using human eyes. With the increasing number of education and sophisticated people in Indian society it is evident that people do not say negative things directly even when they want to say. Generally, an educated and advance mind is more diplomatic than less educated or village people who are not exposed enough to the world. Due to this reason people use more sarcastic language, they say negative things in positive words. Thus, it becomes necessary to identify the true sentiments in the text available on social media or product review or comment pages. In this paper we are demonstrating a system which can help in automatic sarcasm detection in Hinglish language. This work is also extracting text from Hindi twitter handles and Hindi blogs and creating a dataset. The dataset contains the tweets which are written in Roman or Devanagari scripts, words can be from any Indian language or English. In this research no word, either Indian language words written in Roman or English word written in Devanagari is translated or transliterated. A series of data cleaning activities are performed on the text extracted from blogs and twitter. We developed our dataset with the help of 3 Hinglish language speakers. Ten embeddings were built in this work. Four embeddings were finetuned using Transfer embedding, another four embedding were built using training data and standard libraries, one from lexical features and one from lexical features + best embedding. In this work we used ten classification libraries for classification work. In total we developed 109 classification models and analysed the performance of those models against the embedding and classifier used. Four classification models were developed using neural network. Our best model with fastTextWiki embedding and Naïve Bayesian classifier gives 76% accuracy, 78% recall, 75% precision, 76% F1 score and 80% AUC.

[1] https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers (Accessed on 27-Aug-20)

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1. ACC - Accuracy
2. AFINN - Affective dictionary by Finn ˚Arup Nielsen (word with polarity between -5,5)
3. ANEW - Affective Norms for English Word (Emotional Rating)
4. BERT - Bidirectional Encoder Representations from Transformer
5. ALBERT – A lite BERT
6. BN - Bayesian Network
7. Chi - χ Square Test
8. CNN - Convolutional Neural Network
9. CORR - Correlation
10. DAL - Dictionary of Affective Language with degree of these three dimensions Activation, Imagery and Pleasantness .
11. DT - Decision Tree
12. EmoLex - A map between words 8 emotions sadness, joy, disgust, anger, fear, surprise, trust, anticipation
13. EmoSN - EmoSenticNet (IR assigns WordNet Affect emotion labels to SenticNet concepts)
14. EWN - The EffectWordNet lexicon (sense-level lexicon created on the basis of WordNet)
15. GBC - Gradient Boost Classifier
16. GI - The Harvard General Inquirer (Words are labelled with a total of 182 dictionary
17. categories and subcategories + Positive Negative)
18. GR - Gain Ratio
19. GRU - Gated Recurrent Unit

20. HL - The Hu-Liu's lexicon (List of Negative and Positive words)
21. IG - Information Gain
22. KNN - K Nearest Neighbour
23. LFS- Linguistic Features of the Sentence
24. LIWC - Linguistic Inquiry and Word Counts dictionary (psycho-linguistic features in texts.)
25. LMT - Logistic Model Tree
26. LR - Logistic Regression
27. LSTM - Long Short Term Memory
28. MLP - Multilevel Perceptron
29. PMI- Pointwise mutual information
30. POS - Part of Speech
31. RBF - Radial Basis Function
32. RF - Random Forest
33. SCUBA - Sarcasm Classification Using a Behavioural modeling Approach
34. SDS - Sarcasm Detection System
35. SDSHL - Sarcasm Detection System for Hinglish Language
36. SGD - Stochastic Gradient Descent
37. SN - SenticNet
38. SS- SentiSense (It attaches emotional meanings to concepts from the WordNet lexical)
39. SVM - Support Vector Machine
40. SWN - SentiWordNet (word along with POS and sentiment score between 0,1)
41. TIM - Topic-Irony Model
42. TL - Transfer Learning
43. TSTAT - T Statistical Test

CHAPTER 1 : INTRODUCTION

Mobile phones came to India in 1995[2] and Internet was launched in India by VSNL in 1995[3]. Initially the cost of the technology was remarkably high[4], so it was available only to business class, research labs, high level bureaucrats and politicians. With the increase of literacy and decreasing cost of internet services and mobile phone device internet, it is so common that people started thinking that Internet is our fundamental right. As per the World Economic Forum (WEF), in 2019, about 60% of Indian internet users viewed content in vernacular. WEF also says 75% of this 60% is below 35 years of age (Wikipage, 2020b). According to the same Wikipedia page, by 2030, 1.1 billion Indian will have access to Internet and 80% will access the content on mobile devices. The WEF also estimated that 80% of the users will be consuming content in vernacular languages.

When Government of India is going for full blown Digital India program and bringing every citizen of India on the internet platform for purchase, payment and government fund transfer then how the citizens are going to provide feedback about the services which they use? As of today, it is easier to perform sentiment analysis of the feedback given in English, but feedback given in Hindi is not easy to analyse. It means voice of Hindi speaking people is not being considered for service improvement. Till the time somebody is not too angry and do some crime or come on the road to do Dharana or protest we do not know what is happening and why.

Many Hindi news portals, book, blogs, chat bot/WhatsApp conversations, YouTube channels, Twitter & Facebook pages are full of content in Hinglish language. People openly express themselves online using Hinglish language which is mix of Hindi, English, Urdu and other Indian languages. Volume of the online content is increasing at unprecedented rate and it is responsibility of the government, business community, professionals, NGO and accountable

[2] https://en.wikipedia.org/wiki/Telecommunications_in_India#:~:text=In%20August%201995%2C%20then%20Chief,launched%20in%20Kolkata%20in%202012. (Accessed 24-Jun-20)

[3] https://en.wikipedia.org/wiki/Internet_in_India#:~:text=The%20first%20publicly%20available%20internet,not%20permitted%20in%20the%20sector. (Accessed 24-Jun-20)

[4] https://www.news18.com/news/tech/20-years-of-internet-in-india-on-august-15-1995-public-internet-access-was-launched-in-india-1039859.html#:~:text=The%20Gateway%20Internet%20Access%20Service,organisations%20at%209.6%20kbps%20speed. (Accessed 27-Aug-20)

people around to understand the feeling of public and respond accordingly. But the biggest challenge is how to analyse the content which is written in mix of Indian languages. It is impossible to analyse the Hinglish language text manually or using traditional systems.

This section is organized as 1.1. Background of The Study, 1.1.1. What is Hinglish? 1.1.2. Origin of Hinglish, 1.1.3. What is Sarcasm? 1.1.4. Why Sarcasm Detection is Critical? 1.1.5. Why Sarcasm Detection is Critical in Electronic Media? 1.1.6. Sarcasm Detection in Hinglish, 1.1.7. Challenges in Processing Hinglish Language, 1.1.8. Common Challenges in Sarcasm Detection, 1.1.9. Context Understanding A Challenge in Sarcasm Detection, 1.1.10. Challenges in Sarcasm Detection in Hinglish, 1.1.11. Degree of Sarcasm, 1.1.12. Positive Side of Hinglish, 1.2. Problem Statement, 1.3. Aim and Objectives, 1.4. Research Questions, 1.5. Scope of The Study, 1.6. Significance of the Study, 1.6.1. Application of Sarcasm Detection System, 1.6.2. Motivation from Selected Domain

1.1 Background of the Study

1.1.1 What is Hinglish?

There was a time when Hindi was a language which is used by majority of Hindi speaking people when they are communicating (writing, speaking) with each other. But in 21st century, most of the Hindi speaking population who express themselves on social media use Hinglish language. Hinglish is a new lingo of Hindi speaking population. Hinglish sentences follow Hindi grammar and most of the word are taken from Hindi but there is no hesitation of taking words from other languages like English, Urdu, Punjabi, Marathi etc. Hinglish language spoken by different people have different amount of words from different languages. For example, those people who know Urdu good enough for them Hinglish is mix of Hindi, Urdu, English. Those who know Avadhi for them Hinglish is mix of Hindi, Avadhi, English. Those who know Marathi very well for them Hinglish is mix of Hindi, Marathi, English. Thus, in Hinglish Language we have words from Hindi, English and various other Indian languages and written in Devanagari & Roman together.[5] (Sinha and Thakur, 2005) Hindi and English language mixed

[5] Latin is Region and Rome is part of that reason. Over the period of time Roman empire become famous and script was called Roman but Latin is also used simultaneously. https://www.quora.com/Why-is-the-language-of-the-ancient-Romans-called-Latin-and-not-Roman (Accessed 28-Jun-20)

is called Hinglish. Hinglish is not limited to Hindi & English mix, but it includes Punjabi, Gujarati, Marathi, Urdu etc. Phrase construct happens in Roman and Devanagari script.[6]

1.1.2 Origin & Evolution of Hinglish

Before Internet Era in India people use to communicate with each other in much purer format of the language and there was not much mix of other language or English and for writing Hindi they were using Devanagari script. But, with the penetration of internet in the society a new language started taking shape. Initially when Devanagari keyboards were not available people were using Roman letters to write Hindi email, SMS. Like b for ब p for प ph for फ g to ग etc.

An example of late 20th century text in Hinglish language. "Main is doorbhash ka prayog karna nani janta". This is Hindi in Roman script. We need to keep in mind that people do not follow any IAST or other map for writing Hinglish letters in Roman. Mobile phone and Internet were available to elite, educated journalist, professionals. They started realising they are typing in Roman but some words in English so translating them and then typing in Roman is painful. So, text became like this "Main is phone ko use karna nahi janta". Roman script with Hindi and English words.

Over the period of time when Devanagari keyboards were easily available people started using Devanagari keyboards for writing Hindi, but by that time so much English has come in day to day conversation that they felt it is uncomfortable to use Hindi words. So, they write like this. "मैं इस फोन को यूज करना नहीं जानता". Devanagari script with Hindi and English words. Over the period of time people started realizing it is becoming difficult to know which word is Hindi and which one is English therefore a word which come from English root should be written in Roman and word which are from Hindi root should be written in Devanagari. So, they started writing like this. "मैं इस phone को use करना नहीं जानता". Devanagari & Roman mixed for Hindi and English words.

[6] https://en.wikipedia.org/wiki/Hinglish (Accessed 24-Jun-20)

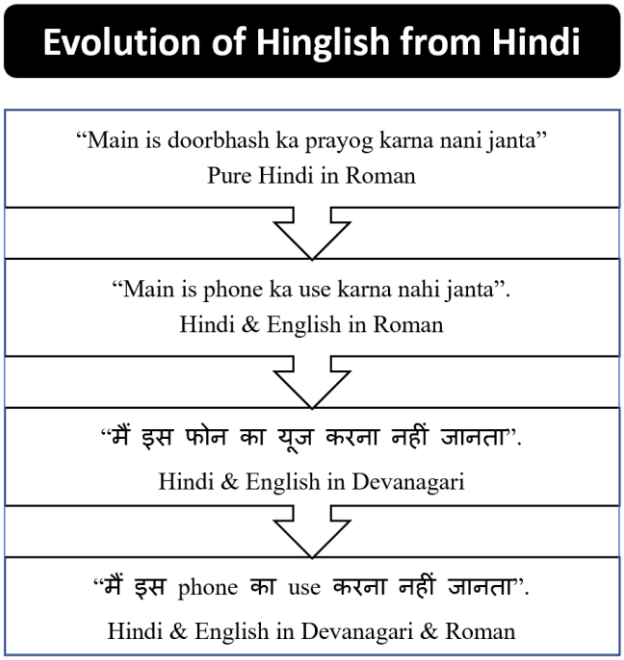


Figure 1: Evolution of Hinglish

Today if you read any Hindi speaker's WhatsApp, twitter or Facebook message you will find they use words from different Indian languages like Urdu, Marathi, Bangla, Punjabi and write either in Devanagari or in Roman. "अमी मोंजुलिका. अमी राजा को जरूर मारबो 😊!, but why you want to kill him?". Here Hindi, Bangla, Urdu and English 4 languages used along with emoticon and written in three scripts Devanagari, Roman & Emoticon. This is Hinglish.

Today Hindi social media, Hindi comment boxes of product, Hindi news articles are full of this kind of language, Hinglish. Therefore, this work using Hinglish language is high value from the angle of practical usage.

1.1.3 What is Sarcasm?

Your friend come to you and speak something to you, from the tone of his language, his body language, choice of his words, time and situation he is speaking you realised that the real meaning of what he is saying is completely opposite. It may be easier for you to detect this opposite sense if you are aware about the complete context but if you are not aware about the context then even as intelligent human you may miss the real meaning of what is being said.

For example, you open the door for your friend, and he says wow! You are looking handsome in this T-shirt. You know that this is an old T-shirt and many times your friend has seen this.

But still not aware of full context, you hesitantly say thank and you invite him inside. After 15 minutes you check yourself in the mirror and realised that you are wearing T-shirt flip side. Now you are embarrassed for your "Thank you" response.

What your friend did was sarcastic remark on your dressing and you being unaware of the full context could not respond properly. In the absence of full context, understanding sarcasm is difficult task and most of the time we take literal meaning of the words or some other time get confused that why someone has made that remarks which was completely out of the context.
In English language this type of grammatical construct which has completely opposite meaning than what is said, it called sarcasm.

As per merriam-webster dictionary, sarcasm is[7]
1: a sharp and often satirical or ironic utterance designed to cut or give pain
2a: a mode of satirical wit depending for its effect on bitter, caustic, and often ironic language that is usually directed against an individual
2b: the use or language of sarcasm

In Hindi it has several name and synonyms like कटाक्ष (Kataksha), तंज (Tanja), व्यंग/ व्यङ्ग (Vyanga), टौंट (Tonta)

Ten forms of humour are *irony, satire, sarcasm, overstatement, self-deprecation, teasing, replies to rhetorical question, clever replies to serious statements, and transformations of frozen expressions*. All these are functions of humour and found in the sitcom (situational comedy). What one finds hilarious or pun may be completely opposite to another person in another country or in other situation. Interpretation is filtered by cultural context. (Anggraini, 2014)

[7] https://www.merriam-webster.com/dictionary/sarcasm

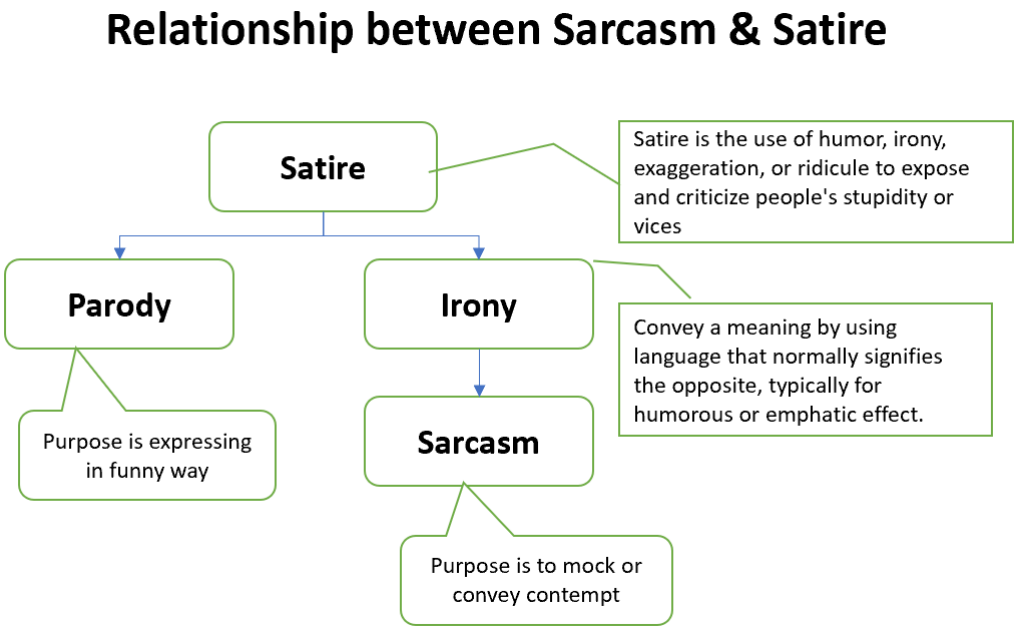


Figure 2: Sarcasm & Satire Relationship

In their work "A Pragmatic Analysis of Humor in Modern Family" (Anggraini, 2014) mentions 11 type of humours. Sarcasm is one type of humour. Let's understand them with example. We are writing examples in English so that English readers can also understand the important of this work.

1- Satire:

Rahul: It looks big accident on the road, let's call police.

Jay: Oh, are you sure? I think police of our state is too busy in catching buffalo of local MLAs.

2- Irony:

Rahul: Why people steal when there are enough opportunities to work hard and earn.

Jay: Oh, you mean those who steal are doing any less hard work?

3- Sarcasm:

Boss: Why do you work so hard, take leave, enjoy life, have some fun after all life is more than work.

Junior: Oh really! Do you know since last one year we are working in Syria? Come with me tomorrow we will go to have fun in a local Jihadi market.

4- Clever replies to serious statements

Rahul: Jay, why didn't you invite me for your birthday party last night?
Jay: I was not sure you will bring any gift for me.

5- Replies to rhetorical questions

Husband: Today is Sunday, why don't you switch off that alarm?
Wife: So that you get up and help me.

6- Teasing

Boyfriend: Where were you when God was distributing brain?
Girlfriend: I was waiting outside for you.

7- Self-deprecation

"They all left the room when I started singing"

8- Overstatement and Understatement Overstatement

Driver: Please pay me 40 dollars for the service.
Passenger: Because of you I missed my flight, your car had problem. First you pay me $500 for the missed flight.

9- Double Entendres

Patient: I am having pain in my right hand.
Doctor: But can you raise your right hand?
Patient: You are nice person, why should I raise my hand before you?

10- Transformations of frozen expression Transformations

“Despite of being hare you are not hearing”

11- Pun

Most people don't use God's most valuable gift to them, their mind. The reason for that is they want to make their God happy by returning His gift as is.

In their work “The Differential Role of Ridicule in Sarcasm and Irony” (Lee et al., 2009) says sarcasm and irony are similar because they are both form of reminder yet they are different because sarcasm is about ridiculing a specific person however this is not required in case of

irony. Sarcasm plays more important role than irony in ridiculing a specific victim. A speaker is more sarcastic when he reminds the listener somebody else's prediction and less sarcastic when he reminds his own mistake.

In our work we will not pay much attention to these specific aspects of humour. Our intention is to detect a sentence which is not carrying the normal meaning. However, most of the records in our dataset which are labelled as sarcastic are sarcastic, but they can have other variation of humour as well.

1.1.4 Why Sarcasm Detection is Critical?

If we do not understand the real intent of the speaker then we cannot respond him properly. Response can be physical action or verbal reply to the speaker or even no action. Sarcasm is like a double edge sword of communication. At one end you can enjoy and another end you can hurt deepest to the opponent. If you do not handle this properly then effect can be completely opposite. Similarly, when other people are sarcastic at us and we are not able to understand the real meaning then other have fun and we ridicule ourselves unknowingly.

Few examples where not understanding the real intent of the person can be catastrophic.

- In face to face communication with your customer when you miss his intent. Result is customer disengagement.
- In live program when you are listening a response or question from the audience in hall or live TV or Radio program or speaking over phone or video conferencing tool and you miss the intent. Result is dent on your reputation.
- In offline communication when you publish some content on blog, news, product selling page and receive some comment from the public. Someone expresses his opinion over your post or tweet, and you are not able to understand that properly or not able to read. All other people read that comment and think that either you are dumb or do not care or accept what is being said. Result you know very well.

When you are dealing with your known people, friends, relatives and not responding properly in that situation, it will have lessor impact because they know your real nature and potential. But in public places, where you do not know the person to whom you need to respond, can cause huge dent on your image and brand.

1.1.5 Why Sarcasm Detection is Critical in Electronic Media?

India a great vibrant democracy so freedom of speech is natural to us. Most of the people in India communicate in Hinglish Language. In democratic societies people have opinion on everything irrespective of their educational qualification and experience. We are a country where public tells how Amitabh Bachchan should act, Virat Kohli should play cricket and how Narendra Modi should run the government. We have view and opinion on everything from politics to religion to product to government functioning to service delivery and what not. Many people choose to remain positive but express their negative feeling in sarcastic way. With the advancement of online sales of products, social media and online blogs, new portals there is huge surge of online feedback. Post COVID19 pandemic there are clear trends of shifting in this direction. People prefer buying, reading, expressing, engaging online. This justifies the need of sophisticated real time sarcasm detection system.

1.1.6 Sarcasm Detection in Hinglish

English[8] is 1st most spoken language in the world and many researchers across the world are working for sarcasm detection in English. But, Hindi is 3rd most spoken language in the world and not much significant work is happening in sarcasm detection in Hindi. Unfortunately, nobody speaks in pure Hindi and it is considered pride unlike English, where people are shamed for not speaking or writing proper English. On social media and public forums a few Hindi speaker use Devanagari to express what they think, mostly they use Hinglish Language. Due to this reason many of the feedback given on twitter, Facebook, product page, online news goes unnoticed and unanalysed.

Sarcasm is one kind of feedback and if we do not use this to improve our response then we prove ourselves foolish and customer shift to different product, service, or platform. Similar things happen when people change their party or group. Therefore, we feel it is extremely important to detect the sarcastic feedback given by those people who write in Hinglish.

1.1.7 Challenges in Processing Hinglish Language

A. Complexity due to English words in Hindi

Observe the variation of a sentence “I have purchased tickets” in Devanagari.

[8] https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers

मैनें (टिकिटें/ टिकटें/ टिकटे/ टिकिट/ टिकट) खरीद (ली/ लीं) (है/हैं). This simple sentence can be spoken in 128 different ways if written in Devanagari. If we mix Roman script in between then number of permutations goes beyond our normal imagination. Here me need to make note that Ticket is English word, and people are making plural of that as they do with any Hindi word.

Let us see another sentence "She has boiled the rice"

उसने राइस बोइल कर दिया है

From the above Hinglish sentence, you cannot figure out whether the doer is female or male, however that is not the case with English sentence. Secondly, राइस and बोइल are not words in Hindi dictionary. Sometime people will write letter in Roman like

उसने Rice बोइल कर दिया है / उसने Rice Boil कर दिया है / उसने राइस Boil कर दिया है /

उसने Rice बोयल कर दिया है

Like Guru, Karma are Hindi words and they are part of English dictionary. We do not have Hinglish dictionary which has word like यूज, गुड, नाइस, क्वीन etc in that dictionary. Without transliterating words like Tickets, Boil into Devanagari and telling system that टिकिटें = टिकटें = टिकटे= टिकिट, बोइल= बोयल our embedding will not give good.

B. Mix Other Indian Language with Hindi

Observe the sentence below, Bangla written in Devanagari and clearly understandable by any Hindi speaking person. Most of the words in the sentence below are from Bangla language but written in Devanagari.

अमी मोंजुलिका.अमी राजा को मारबो दीदी ने केजरीवाल को भी पीछे छोड़ दिया. जि तो कमालई कर दओ दद्दू

India's business film Industry in Mumbai make film in Hindi. Rarely any film use as good Hindi as Hollywood uses English. Adoption of words from other language is not a problem. The problem is quantity of the words taken from other languages and non-availability of the updated vocabulary of the language. Many famous dialogues or songs from Hindi films which are taken from different language or dialects. This increases complexity of sarcasm detection in Hinglish. We do not have comprehensive dictionary which we can call Hinglish dictionary which has all the word being used by the Hinglish speakers.

Without telling system that अमी (Bangla word) = मैं, मारोबो (Bangla word) = मारूंगी = मारूंगा = मारना no embedding is going to help

C. Complexity of Synonyms in Hindi

For this let us understand what Synonyms is. A word or phrase that means exactly or nearly the same as another word or phrase in the same language[9], for example "shut" is a synonym of "close". Few examples of synonyms

- The East = The Soviet Union (https://www.lexico.com/en/definition/synonym)
- Country of rising sun = Japan, Dragon Country = China,
- Fridge = Refrigerator
- Happy = Joyful, Cheerful, Contented, Jolly, Gleeful, Carefree

In the case of Hindi, it is very much different.

D. Influence of Sanskrit

All the synonyms have different spelling, different pronunciation but almost same meaning and part of the same language. l'eau (French word for water) is not synonyms of water because they are two different languages.

Unlike other world languages, all Indian languages (except Tamil, this is debatable) heavily borrow words from Sanskrit.

Let's take English word "Water" and see how many words are available in sanskrit for "water" जल = पानी = तनि = नीरू = आपः = वाः = वारि = सलिलं = पयः = तोयं = मेघपुष्पं = घनरसः = पाणी. So all these words are synonyms of water in sanskrit.

Because all Indian languages have root in Sanskrit therefore most of the time, they take word from Sanskrit for communication. For example, Kannada uses नीरू, Bangla use पानी, Hindi uses पानी, सलिलं, मेघपुष्पं. Even if not used regularly, they are used in poetical or sometimes in sarcastic language. Because in sarcasm or poetry we often use loaded words.

[9] https://www.lexico.com/en/definition/synonym.

In Hindi language, can we say नीरू is synonym of पानी? No, because नीरू word normally is used in Kannada and Sanskrit and not in Hindi. As per the definition of synonym another equal word should be from the same language and we know Hindi is not Kannada nor it is Sanksrit. The answer is yes also; because Sanskrit being mother of Hindi language, it borrows words freely from Sanskrit. Thus, we see synonym in Hinglish is not the way it is understood in the context of English.

Therefore, to be build a complete Hinglish dictionary we have to take words from all other Indian languages and frequently used English words as well. Thus, it should be like this.

जल = पानी = तनि = नीरू = आपः = वाः = वारि = सलिलं = पयः = तोयं = मेघपुष्पं = घनरसः = वाटर

E. Variation in Spelling of Same Word

In Hindi same word spoken and written with different spelling. Observe the spelling of the same word how they are varying. This kind of problem we do not have in English. As discussed earlier, synonym of Happy is Jolly. They both are not same, neither in spelling, nor in pronunciation, nor in full sense, but "happy" is close to "jolly". That is why they are synonyms. But below all "=" signs are referring to the same thing.

विष्णु = बिश्णु = विश्णु = बिष्णु = विष्नु = बिष्नु,

दरसन= दर्शन= दर्सन = दरशन

करता = कर्ता,

यज्ञ = जग्य,

योग = जोग,

हरि=हरी,

We need to keep in mind Hindi is not Devanagari, nor Hindi is Avadhi or Marathi. Hindi is written in Devanagari script, but it is heavily inflicted by other languages like Awadhi, Bhojpuri, Rajasthani, Urdu etc.

Unless we have a dictionary, which tells विष्णु = बिश्णु = विश्णु = बिष्णु = विष्नु = बिष्नु, embedding will not help.

1.1.8 Common Challenges in Sarcasm Detection

Detecting Sarcasm is difficult if sentences are having following characteristics.

A. **Idioms and Phrases**: Sarcasm detection become more difficult when people speak in idiomatic language. For example: “What a wise man! what he did is nothing other than an axe to grind.” “कितना समझदार आदमी है जो उसने किया वो अपने पैर पर कुल्हाडी मारने के सिवा कुछ और नहीं है”

B. **Speaking with Hint:** When people do not talk directly and use examples which are completely different than context. For example: “You are behaving like Mir Jafar.” “तुम्हारा व्यवहार मीर जाफर जैसा है”

C. **Culture:** Different languages have different degree of challenges in sarcasm detection. For example, English is spoken all over the world but the way American express their feeling is different than the way British express. The reason for that is the work and social culture of England and United States is hugely different. In English language what is called sarcasm in England may be considered a normal statement or abusive in US and vice versa.

D. **Datasource:** Sarcasm can be present in any kind of communication platform like WhatsApp, twitter, Facebook, reddit, LinkedIn, product review, movie review, news review, blog review etc. But, because of the type of audience, type of input interface, awareness of topic, command over language, character limit, text formatting possibility etc content available on the various platform has different characteristics. For example, twitter content is short and full of acronyms, words without vowel, scripting language mixed. On the other hand whatsapp group communications are full of links, emoticons and forwards with little text written by sender.

1.1.9 Context Understanding a Challenge in Sarcasm Detection

Since the time human child take birth, baby has environment to learn from. Various types of formal or informal environment, social or business or cultural background forces human to think and learn. Either at physical or emotional or intellectual level if human fail to learn then his survival is challenged by the nature around. In this kind of environment, it is easy for any human to understand the context. If we are alert and interested in the topic then we need not to struggle hard to understand the context. But context understanding is extremely difficult in the

case of Machine learning. Let us analyze one sarcastic tweet. "#JIO का सच नीता अंबानी ने मन्नत मांगी थी कि अनंत अम्बानी अपना वजन कम कर लेगा तो गरीबों में 3 महीने Net or call का भंडारा करवाऊँगी"

People living in India can understand that this is sarcasm. Because we know the full context. That

- Mukesh Ambani is owner of #Jio
- Neeta Ambani is Wife of Mukesh Ambani
- Anant Ambani is son of Neeta Ambani
- Anant Ambani has 200+ Kg body weight
- Normal body weight of human is around 70 kg
- Anant Ambani is overweight as per the normal standard
- Neeta Ambani desired that her son should have normal weight
- #Jio has launched 3 Month free internet package
- There is no direct connection between Anant Ambani weight reduction and 3-month free internet package

(Joshi et al., 2018) in their work "Investigation on Computational Sarcasm" says there are three type of context, Author Specific context, Conversational Context, Topical Context

We need to understand that keeping all these facts in mind we can say a statement is sarcasm and not a normal statement. Even a human, who does not have all this information will fail to classify a statement as sarcasm. It is not easy to give all this information to a system to make a classification decision

1.1.10 Challenges in Sarcasm Detection in Hinglish

A. Script used for writing

70% of the world population uses 26 letters of Roman script to write their language. The Roman alphabet is also used as the basis for the International Phonetic Alphabet, which is used to express the phonetics of all languages.[10] Due to this reason when people are writing different language like English, French, Indonesian, Tagalog, German, Turkish they need not to change much around the letters, so most of the cases script remain Roman. This advantage is not available to Devanagari script and Hindi language.

[10] https://www.worldatlas.com/articles/the-world-s-most-popular-writing-scripts.html Accessed on 23-Jun-20

“Badhai ho kongressi Pappu ki vajah se #मोदी चुनाव फिर जीत गये” This entire sentence is in Hindi but notice script used is Devanagari and Roman. Not only that note the spelling of “congress”. Because this is how native speaker think when he thinks about the sound of “क” or “K”.

While typing feedback people write @account_name. Most of the time @account_name are proper name and written in Roman like @harithapliyal, @eating_point, @banarasi. Similarly, hashtag, which helps us understanding the context of the feedback, is also written in Roman script #Election2019 #COVID19 #Philosopy #Motivation #NarendraModi.

B. Language mixed

An average westerner knows and speaks one language so written and verbal expression most of the time is that one language. An average educated Indian speaks minimum 3 languages, one is language of his state/community/region, seond national language and third is English. In southern part of India, it is not uncommon when you find a taxi or truck driver who can speaker 3 or 4 languages, but they cannot speak in English. This, one language- one script, advantage is not available for any Indian and they communicate in multiple language without realising that they have shifted language and borrowing words from different language.

“रहने दो उसको, उसके food preparation speed itna fast hai ki जितनी देर में राजधानी रेस्टरां वाले खाना घर पर डिलिवरी कर जायेंगे” This is sarcastic sentence about the laziness of the other person.

But analyze the words and language this

“रहने दो उसको, उसके” script Devanagari, language Hindi

“food preparation speed” script Roman, language English

“itna ---- hai ki” script Roman, language Hindi

“रेस्टरां, डिलिवरी” script Devanagari, language English

No matter how big corpora we use for tokenization and embedding, what kind of technique we use for tokenization till we have this kind of mix corpora for training sarcasm prediction in these kind of sentences is always going to be challenging.

C. Missing Context

“I love working hard” It looks normal sentence. But, if you add a context “my brother trying to still sleeping at 9am and saying” then meaning of the original statement is not what the speaker it saying. Thus, the missing context or context not fully defined lead to issues of sarcasm detection in the sentences.

D. **Limitation of Written Languages**

Let’s take one sentence “I didn’t say he beats his wife”. It is simple statement by the speaker, where he is making a point about what he knows. But how it is understood also depends in what tone it is said. If he emphasis on “his” then it looks like “I didn’t he beats HIS wife” it can imply that he beats but not his wife. Written language has its own limitation. Message may not be expressed properly and tone of speech, body language, eye contact, facial expression etc which art part of audio-visual domain of communication has lot hidden in it. So, the message still may be sarcastic, but it is not part of the written words.

E. Usage of Idioms & Phrases

आ गया ऊंट पहाड़ के नीचे?

There is nothing special in the words of this sentence. But this is idiomatic phrase, and you use it in some context and with interrogation marks then it is sarcasm on someone. It is not easy to know whether sentence contains idiomatic phrase or normal phrase.

F. Sentences containing Emoticon, Interjections etc.

अरे वा! इनको इस महान कार्य के लिऐ तो कम से पद्मश्री award मिलना ही चाहिऐ. 😊 🤣

This looks normal sentence but emoticon and interjection is sarcastic

ओ साहेब, क्या समझ रखा है इतनी मेहनत के बात पद्मश्री award नहीं labour मजदूरी मांग रहे है ☹

This second sentence also has emoticons and interjections, but it is not sarcastic. It is challenging task to comprehend the meaning that too when text is mixed with emoticon and interjections.

G. Different Numerals

Many times, people use non-English numerals like १, २, ३, ४, ५. Depending upon the regional language people use different numerals for writing the same numbers.

A detail report on Transliteration challenges in Hinglish Language is available at github.

1.1.11 Degree of sarcasm

Although how a person perceive & responds to a sarcasm it also depends upon him, yet we need to know all sarcastic statements are not equally intense or powerful to generate pain to the listener or reader. Here are few examples of different degree of sarcasm.

A. ओ भाई कचोरी समोसे की दुकानें खुल तो गयी है लेकिन ध्यान रखे कचोरी समोसे के चक्कर में आप की ही पूड़ी सब्जी न बट जाये #Covid_Unlock (Least Intense)

B. NDTV की हैडलाइन एक बेजुबान अल्पसंख्यक भैंस को डूबा कर मारने की कोशिश करती बहुसंख्यक चिड़िया (Lessor intensity)

C. करोना का दवा न होना यह एक साइंस है, और दवा न होते हुए भी बिल लाखों मे आना ये एक आर्ट है !! (Moderate Intensity)

D. ये शुक्र है जंगल में आरक्षण नहीं, बहोत नहीं तो जंगल का राजा शेर नहीं गधा होता. आरक्षण खत्म करो 70 साल हो गये यार #आरक्षण_भीख_है (Sharp Intensity)

1.1.12 Positive Side of Hinglish

Although India is big country with 1.35 billion people with different culture, religion, tradition but there is some common aspect in India culture and this does not change no matter where an Indian is living on the earth. That common culture helps us understanding the context and intent easily. Although there are many languages in India but because of one overarching culture it is easier to understand the meaning, a simple translation is good enough. Unlike English where Australian struggle to understand what American gentlemen want to say in English.

1.2 Problem Statement

More than 4.5 billion people now use the internet, while social media is used by approximately 3.8 billion users. Nearly 60 percent of the world's population is already online, and the recent trends highlighting that more than fifty percent of the world's total population will use social media by the middle of 2020.[11] IT companies like google, Facebook, twitter, amazon, Alibaba, Linkedin, Instagram, Quora dominate the content on Internet.

[11] https://wearesocial.com/blog/2020/01/digital-2020-3-8-billion-people-use-social-media#:~:text=More%20than%204.5%20billion%20people,the%20middle%20of%20this%20year. (09-Oct-20)

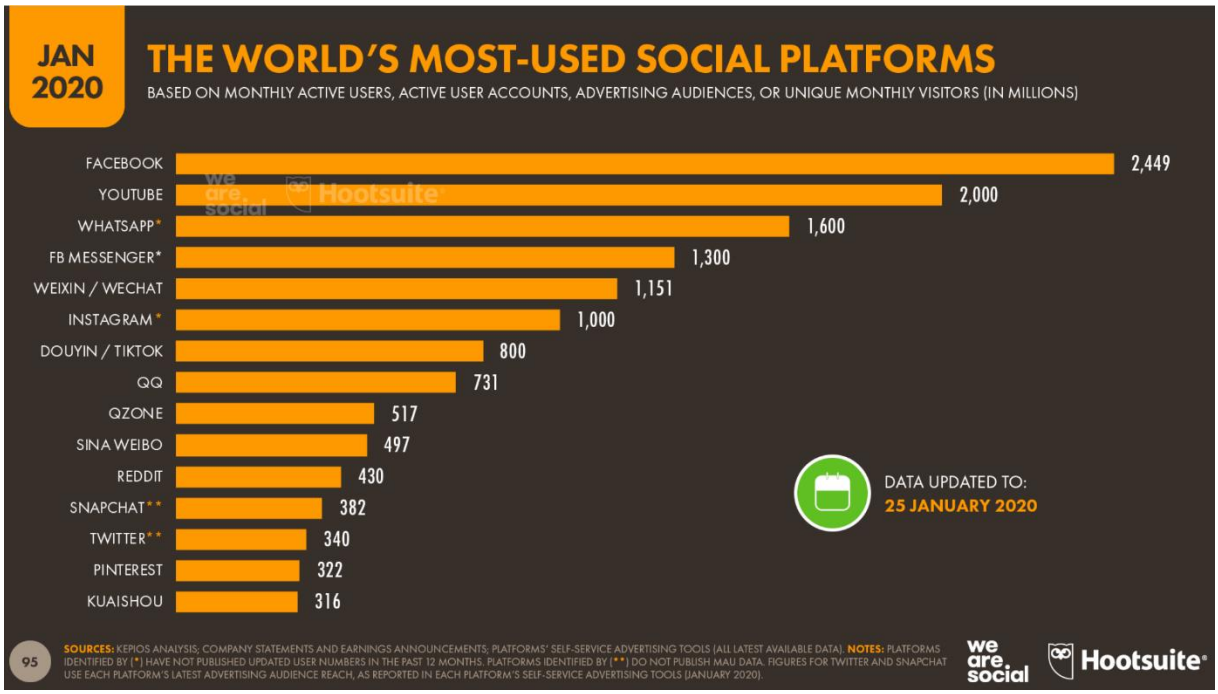


Figure 3: Usage of Social Media Platforms

Keeping this volume, demand and need in mind, we want to develop a sarcasm detection system for Hinglish language which can work for all social media content, reviews, comments, and feedbacks.

1.3 Aim and Objectives

The aim of this research is to propose a model, which can predict sarcasm in a given Hinglish language sentence with highest possible accuracy.

Based on the above primary goal, objectives of this research are as following.

A. To create Hinglish language dataset with minimum 2000 sentences, which can be used for training and testing a sarcasm detection model of Hinglish Language
B. To develop a sarcasm detection models
C. To check the effectiveness of Transfer learning for our work.
D. To understand which embedding model or library works best for Hinglish language.

1.4 Research Questions

A. To study how sarcasm detection is done by other researchers for English or any other Indian languages?
B. To determine which word embedding & linguistic features works best for sarcasm detection in our Hinglish dataset?

C. How to do transliteration from Roman to Devanagari? Many options are available for reverse translation. For example "एकीकरण" => "Ekikaran" is easy and many options are there but "Ekikaran" => "एकीकरण" is not easy. Because Hindi speaking population is not aware about IAST[12] and nor they use it for transliteration. So confusion is how to convert word of Roman script to Devanagari, for example (a) "ra"=> र or ऱ, (b) n=> न or न् or ण or ण् or ञ or ञ् or ङ or ङ्, (c) ki=> कि or की or क्ि or क्ी or क् इ or क् ई

D. Is transfer learning useful for our work?

1.5 Scope of the Study

A. This research is not related to any specific domain like philosophy, politics, history, current affair new etc. Rather it is trying to detect sarcasm in day to day informal conversation.

B. Sarcasm in our communication can be expressed and experienced at Visual (facial express, body language), Vocal (tone, pace of speech, emphasis on certain word) and text (book, newspaper, articles, social media tweets, comments and feedback box on internet. Visual sarcasm is more universal than vocal. Because voice uses language and there are 7000+ languages on the earth so there is no universal vocal language of expressing sarcasm. But pause, pitch, pace, modulation between words, while speaking, are more universal like Visual. In this paper we are deal only with text-based sarcasm.

C. Only Roman and Devanagari scripts are considered.

D. Only Hindi and English language words are considered. If we find sentence using words from other languages, then we will drop those sentences from our dataset.

E. No analysis of degree of sarcasm.

F. We know to understand the context datetime plays a critical role. Our base dataset does not have datetime. And lots of the text in the dataset is coming from non-tweet sources which does not have datetime chronology of communication. Therefore, we ignored context which is coming from datetime. We want our system to be indifferent of datetime metatag.

1.6 Significance of the Study

We did not find any one place which claims that we have done research and can say with conviction that approximately these are the number of Hindi speaker in the world. Different

[12] https://en.wikipedia.org/wiki/International_Alphabet_of_Sanskrit_Transliteration

sources reveal different numbers. As per a lingoda.com[13] and babbel.com[14] after English and Mandarin Hindi is 3rd most spoken language on earth. It is spoken by 615mn people. As per Wikipedia 176 million people speak Urdu.[15]

Culture of Hindi speaking population and Urdu speaking population resembles a lot. While speaking or writing Hinglish many words of Urdu are spoken or written unknowingly. Therefore, any sarcasm analysis system in Hinglish will benefit Urdu speaking community as well.

With current trend of increasing online content in Hindi, it is practically not possible to read every review, even if you try it is very expensive and not worth work. We know, even one negative feedback or abuse which goes unnoticed can cause huge problem for the brand of the company, product, or person. Therefore, performing sentiment analysis on every feedback makes a perfect sense and it can be done automatically almost in real time.

Sarcasm is one type of sentiment and we are trying to discuss overall benefits of sentiment analysis keeping Sarcasm at the centre of discussion.

1.6.1 Application of Sarcasm Detection System

A. Sarcasm analysis is one kind of Sentiment analysis. Sentiment analysis has a broad range of applications like understanding whether a feedback is Sarcasm, Warning, Love Emotion, Hate Emotion, Advertisement of some other product, Contradicting statement, Pun, Abuse, Inspiring Quote, Sensational Revelation, Pleasant Surprise, Allegation, Poetry/Dohe/Chands etc.
B. Government, NGO, religious leaders, product sellers are able to perform the sarcasm analysis against some product, political party, ideology, religion, company etc. then they will be able to control the situation in much better way with minimum damage.
C. Sarcasm analysis can be used to analyse the feedback on airlines service, travel service, bus or taxi service, telecom, health, government service, new articles, personal blog,

[13] https://blog.lingoda.com/en/most-spoken-languages-in-the-world-in-2020 Accessed on 22-Jun-20

[14] https://www.babbel.com/en/magazine/the-10-most-spoken-languages-in-the-world Accessed on 22-Jun-20

[15] https://en.wikipedia.org/wiki/Urdu Accessed on 22-Jun-20

food delivery, insurance service, personality page, book page are good places where sentiment analysis plays a critical role.

D. In multinational companies it becomes exceedingly difficult to use humour to communicate the idea, crack joke or sarcasm, even if all the team member can speak English. The reason for that is different cultural background and different level of comprehension of English by non-native speakers. But when Hindi speaking people connect over video, telephonic or chat conversation it is easy for them to use idioms, joke, sarcasm and ensure that idea is understood. There is different kind of joy of working in lesser formal and light-hearted environment. When Indian people are speaking to each other using Hinglish we can perform sarcasm analysis to know the feeling of the group.

1.6.2 Motivation from Selected Domains

Below are examples of motivation written in English language. We have taken examples of sarcasm enabled chatbots. Answers given below by a chatbot is possible only if chatbot can understand that input given is sarcasm and nor normal text.

A. Motivation in Travel Domain

Passenger: #ac_not_working. I love to get roasted in heat.

Chatbot: Sorry for the inconvenience. Our service engineer will call you.

B. Motivation in Hospital Business

Attendant: #expensive_treatment. We come to your hospital for this expensive treatment so that we can talk to your cute nurses.

Chatbot: We understand your concern about treatment cost. Our billing manager will call you.

C. Motivation in Restaurant Business

Customer: Last time, your food was so good that since last 2 days I am taking rest.

Chabot: I am sorry to hear that.

D. Motivation in Learning Portal

Learner: What a great content. Since last 30 minutes I am still trying to understand the head and tail of that 30 minuite video.

Chatbot: Sorry, can you please share with us what difficulty you faced?

E. Motivation in News Portal

Reader: What a great story! Did you read it after writing?

Chatbot: We are sorry that you did not like this story.

F. Motivation in Airlines Business

Traveler: First time in my life I got such a wonderful service from any airlines. I reached to the destination one day before my check-in baggage.

Chatbot: We are sorry to hear that. We hope your baggage reached safe to you.

G. Motivation in Dialogue Analysis Work

A dialogue from a Hindi Film "Sholey"[16]

मौसी मेरा दोस्त इतना अच्छा है कि वह शराब को कभी न नहीं बोल पाता। पीने के बाद जुआ खेलना उसकी खूबी है इसमें उसका कोई दोष थोडी है मौसी। बस हारने के बाद थोडा मारपीट करता है और घर में आ के मेरे को गाली देता है। पर मेरा दोस्त दिल का बहुत अच्छा है मौसी आप अपनी बेटी की शादी मेरे दोस्त से पक्की कर दो

This is a pure sarcasm paragraph. These kind of dialogues makes movie interesting.

1.7 Structure of the Study

Structure of the study is as following.

2.1. Sarcasm Detection Systems (SDS)

2.2. History of Sarcasm Detection Systems

2.3. General Purpose Text-Based Sarcasm Detection System

2.4. Feature Engineering In Sarcasm Detection Systems

2.5. Approaches to Develop SDS

2.6. Approaches to Handle Key Challenges in Sarcasm Detection

2.7. Embedding

2.8. Types of Sarcasm Detection System

[16] https://en.wikipedia.org/wiki/Sholay

CHAPTER 2 : LITERATURE REVIEW

Lot of work has been done in English language sarcasm detection and authors mentioned different challenges in sarcasm detection, although results are not that great as for any other classification or other sentiment analysis problems. Challenges exists because of context understanding, missing context, domain, culture, different words, or expression used by people to flip the meaning etc. There is not much work done in Hinglish Language Sarcasm detection. Hinglish language has a separate set of challenge like mixing script, mixing language, highly morphological words, using same morphology on English language words, meagre size of corpus etc.

Let us take one English verb "do", in Hindi, it can be used like कर्ता (noun) , करता (verb with male), करती (verb with female), करूंगा (future tense with male), करूंगी (future tense with female), करेंगे(future tense with plural), किया (did, done), करो (request, must do) करें (please do) etc. these all are with different gender, mood and tenses. However, in English we have infliction like do, does, did, done. These inflictions in Hindi are such that even without using pronoun sentence is meaningful. For example, करता है = वह करता है. Even without pronoun वह sentence is correct, complete, and meaningful. While this is not true in the case of English language.

Now, let's take another example but this time we take noun "Ram". राम का, राम ने, राम को, राम द्वारा, राम में, राम पर, राम के लिए, राम पर and many times you will see letters are written together. We never see any word like "ByRam" in English but in Hindi रामने and राम ने both have same meaning. In sanskrit we call it Vibhakti (विभक्ती)

2.1 Sarcasm Detection Systems (SDS)

Sarcasm is perception of the human receiver about some inputs. "Input" can be of four types. First type of input is text format written in social media, book, newspaper etc. Second type of input can be vocal tone, expressed in some voice communication over phone, face to face meeting, stage show, etc. Third kind of input can be image appearing on some public hoarding, newspaper article, blog post, social media etc. Forth kind of input can be body language of human during face to face interaction or in video.

To understand a message correctly following conditions should be met successfully.

- Speaker speaks in the language which listener can understand
- Listener understand the background
- Listener has technical knowledge about the subject

Beauty is in the eye of beholder. If receiver missed the sarcastic intent of input due to any reason, then will you call that statement sarcastic? This is philosophical debate and, in our work, we will be focusing on text which is marked as sarcastic by different annotators. From receiver's perspective input received can be any of the following four types.

All Weather Sarcastic: Every civilized person will treat those statements as sarcastic. For example, "I like when you treat me like a slave". No matter what the context is, what language is used to communicate this text everybody will say this is sarcastic statement. No other information is required, sentence has complete information and almost all human agree to this.

Conditionally Sarcastic I: More information is required to classify a sentence is sarcastic or not. In the presence of that important information we can confidently say this is normal or sarcastic sentences. This more information may be related to profession, culture, rules, law of the land etc. For example, "I love to beat drum at 5 am in the morning". Some cultures, profession forces their follower, community members to do eating, praying, singing, playing activities at a time so in that profession or community's context it may be normal. Otherwise it is sarcastic.

Conditionally Sarcastic II: Sometimes we need *individual event and person specific information* to detect sarcasm in sentence and this information cannot be generalized even for the same person at other time. For example "First thing in morning I like to do is cleaning potty of Ruby". If receiver know the context that the speaker is mother and Ruby is new born baby then speaker may say it is sarcastic or non-sarcastic depends upon receivers individual like or dislike. Here even after understanding the full context it is depending upon the receiver who does the classification. But if receiver knows that that speaker is busy CXO and Ruby is his pet name. Then receiver will say it is definitely a sarcastic statement.

Non-Sarcastic: Normal sentences with straight forward meaning without hiding any intention and no scope of different interpretation.

Sarcasm detection system is one which can flag an "input" provided to the system as sarcastic or non-sarcastic. In the context of our project "input" mean text and no other type of input like body language, image, video, speech etc. Even with text as "input" we are particularly dealing with one or two liner text appearing on social media or day to day communication. We are not dealing with long chain of text like a paragraphs, a page, a chapter or a book. We are interested in developing state of art sarcasm detection system for Hinglish language. Systems takes input as one or two sentences with full context and returns True or 1 if Hinglish sentence is sarcastic else returns false or 0. If the context is missing, then system may fail to predict correctly.

2.2 History of Sarcasm Detection Systems

We have prepared a separate report on History of Sarcasm Detection. If you are interested in the chronology of the development you can check it from github link. For the purpose of brevity we have kept this report outside of this work.

2.3 Generic Text-based Sarcasm Detection System (GTSDS)

There are many dimensions of complexity in any sarcasm detection. General purpose text-based sarcasm detection system means a system which can detect sarcasm in any text. Before building a GTSDC we need to answer following question.

- Can we develop one SDS which can detect sarcasm expressed in all human language like English, Hindi, Japanese, Chinese, Spanish etc.?
- Can we develop one SDS where text written in any script like Devanagari, Roman, Hebrew, Chinese etc. can be identified?
- Can we develop one SDS which can detect sarcasm from text, written in simple language vs figurative language which is full of proverbs and coded words?
- Can we develop one SDS which can detect sarcasm from the text, which is using words from any business domain like politics, philosophy, medical practitioners, lawyers etc.?
- Can we develop a SDS which can detect sarcasm from the text written by the people of different culture like British, North American, Indian, Japanese etc?

Building a general-purpose text-based sarcasm detection is extremely complex task. In this work we are trying to develop a general purpose SDS where the text of two scripts and words from multiple language like Hindi, Sanskrit, Urdu, Punabi, Marathi, Bhojpuri, Avadhi are used. We are aware this is not a complete *generic purpose text-based sarcasm detection system* and but a small step towards that.

2.4 Feature Engineering in Sarcasm Detection Systems

Researchers have extracted various features from the given text to detect whether sentence is sarcastic or not. These features can be grouped under following categories.

- **Lexical**: unigram, bigram. These can be created using words or characters.
- **Pragmatic**: These features are created using emoticons, punctuations and capital letters used
- **Incongruity**: Incongruity in the sentences is detected using novel approaches.
- **Polarity**: Polarity of the noun, adverb, adjectives are counted
- **Syntactical**: These features are based on POS (part of speech)
- **Idiosyncratic**: Sentences are analysed for repetition of any specific word by the speaker. Many people have habit of say words like "I know", "you know", "yah yah", "absolutely", "like" etc.
- **Prosodic**: Analysing pattern of words in the sentence, how a specific words is written to emphasis something. For example "It is soooooooooooooo beauuuuuuuuuuuuuutiful"
- **Features based on the Author's or reader's profile data**: Gender, nationality, religion, education, ideology, familiarity of language etc.
- **Features based on the environment**: Datetime, current news, messages in past, present state of mind etc.
- **Hashtag & @users**: Different hashtags used and different users tagged in the message
- **Slang**: Number of slang used, ratio of slag to normal words, nature of slang word etc.
- **Profanity**: Any dirty, abusive, naughty, offensive words

There are many creative ways to create hundreds of features under above categories. We will refer all these features as Linguistic Features of the Sentence (LFS)

In their work, (Joshi et al., 2018) have used 3 types of features POS, Named Entities, Unigram to predict the disagreement. (Sharma et al., 2014) in their work "A Sentiment Analyzer for Hindi Using Hindi Senti Lexicon" suggests using bootstrap approach to extract sentiment words from Hindi Wordnet. It has given encouraging results of 87% accuracy in sentiment analysis. We are going to test usefulness of this approach in sarcasm detection.

We have prepared a "Summary of Papers on Sarcasm Detection". This presents a summary of these features used by different researchers and the performance reported by them. If you interested to read more, you can refer to the github repository.

2.5 Approaches to Develop SDS

Over the period of last 20 years different approaches are adopted by different researchers. Broadly these can be categorized into following categories. In the following subsections we are analyzing features explored, algorithms used, and results gained by the different researchers. If you want to more about these then you can refer to our work "Summary of Papers on Sarcasm Detection" and History of Sarcasm Detection. Table below presents the summary of approaches used to develop SDS. Numbers written in the cells of the table are section number following the table.

Classification Type - Feature Type
Discussed in Section Number

		Feature Types		
		LFS	Embedding	Both
Classification Type	Rule Based	2.5.1	x	x
	Classical ML Algorithms	2.5.2	2.5.3	2.5.4
	CNN	2.5.5	2.5.6	2.5.7
	Transformers	x	2.5.8	x
	Transfer Learning	x	2.5.9	x

Table 1: Classification Types & Feature Types Mapping

2.5.1 Rule based Approaches

In this approach researcher depends upon the content and context-based of the text. They extracted various Linguistic Features of the Sentence (LFS). Some experimenters have demonstrated a good performance on sarcasm detection work without using any machine

learning algorithm. “Lexicon-Based Sentiment Analysis in the Social Web” by (Asghar et al., 2014) didn’t use any classical or neural network based algorithm for this work. They could achieve 95% accuracy by using a) Lexical features- unigram using chi-square test, (b) Pragmatic- emoticons, punctuation marks, capital words, (c) Explicit congruity- related to polarity changed, and (d) Implicit incongruity features.

Just using rule based approaches (Bharti et al., 2018) achieved 87% accuracy on Hindi language tweets and (Sharma et al., 2014) could achieve 85-89.5% accuracy on Hindi language product reviews.

2.5.2 Classical Machine Learning with Linguistic Features

In this approach we can use LFS but classification is done using the classification machine learning algorithms like LR, SVM, RF etc. Many experiments are done using this approach.

(Fańias et al., 2016) demonstrated 73-96% accuracy using classifiers like SVM, DT, NB and feature engineering approaches. (Suhaimin et al., 2017) shown 82.5% accuracy with non-linear SVM. Both of these experiments are done on English tweets.

(Sundararajan and Palanisamy, 2020) used English twitter data and shown 86.61% to 99.79% accuracy using classifiers like Random Forest, Naive Bayes, Support Vector Machine, K-Nearest Neighbor, Gradient Boosting, AdaBoost, Logistic Regression, and Decision Tree. They extracted 20 features from the dataset.

In an another interesting work on Instagram image (English text), (Kumar and Garg, 2019) has developed a sarcasm detection system with 73% to 88% accuracy. They extracted features like Number of negative words, number of positive words, POS tag, hashtag from the dataset.

2.5.3 Classical Machine Learning with Embedding

In this approach we need not to explore any LFS. Using word embedding word vectors are created and they can be used for creating classification model. During literature survey we could not find any papers which solely rely on word embedding for creating the models.

2.5.4 Classical Machine Learning with Embedding & Other Features

Using this approach, we create LFS along with word embedding for every sentence. (Kumar et al., 2019) used tokens using Classical language toolkit, unigram, bigram. They also used fastText and TF-IDF embedding. Authors used SVM linear kernel, LR, RF, Shallow CNN + Bi-Directional LSTM for classification purpose. In their work "BHAAV- A Text Corpus for Emotion Analysis from Hindi Stories" they were trying to classify emotions in Hindi language sentences. They claim that they could get an accuracy of 62%.

2.5.5 Deep Learning with Linguistic Features

In this approach deep learning neural networks are explored for classification but instead of using word embedding Linguistic features of the sentence are used. (Liu et al., 2019a) in their work "A2Text-net: A novel deep neural network for sarcasm detection" used CNN and created a novel architecture for sarcasm detection. They tested their model on different dataset and got different results. The results vary between 71%-90% F1 score.

2.5.6 Deep Learning with Word Embedding

Deep learning approaches includes those experiments where experimenters have used CNN, RNN, GRU, LSTM or any variation of neural network. They transformed the text input into vectors using different embedding techniques like TF-IFD, word2vec, fastText etc. (Subramanian et al., 2019) used GRU on English language twitter and facebook dataset and show 89.36% accuracy on twitter dataset and 97.97% accuracy on facebook dataset.

2.5.7 Deep Learning with Embedding & Other Features

In this approach, researchers created features using both the word embedding and LFS. For classification researchers used deep learning networks like CNN. In "CARER: Contextualized Affect Representations for Emotion Recognition" (Saravia et al., 2020) used BoW, char n-gram, TF-IDF, Word2Vec, fastText(ch), word-cluster, enriched patterns, Twitter-based pre-trained word embeddings and reweight them via a sentiment corpus through distant supervision. Authors used CNN for the classification and claimed an accuracy of 81% using their novel architecture named CARER.

2.5.8 Transformer Based

(Vaswani et al., 2017) in their work "Attention Is All You Need" proposed a novel architecture which named Transformer Model Architecture. A transformer has two units first is encoder,

and second is decoder. Subcomponents of transformer architecture are positional encoding, multi-headed attention, feed forward network, masked multi-headed attention, fully connected dense layer and finally Softmax layer.

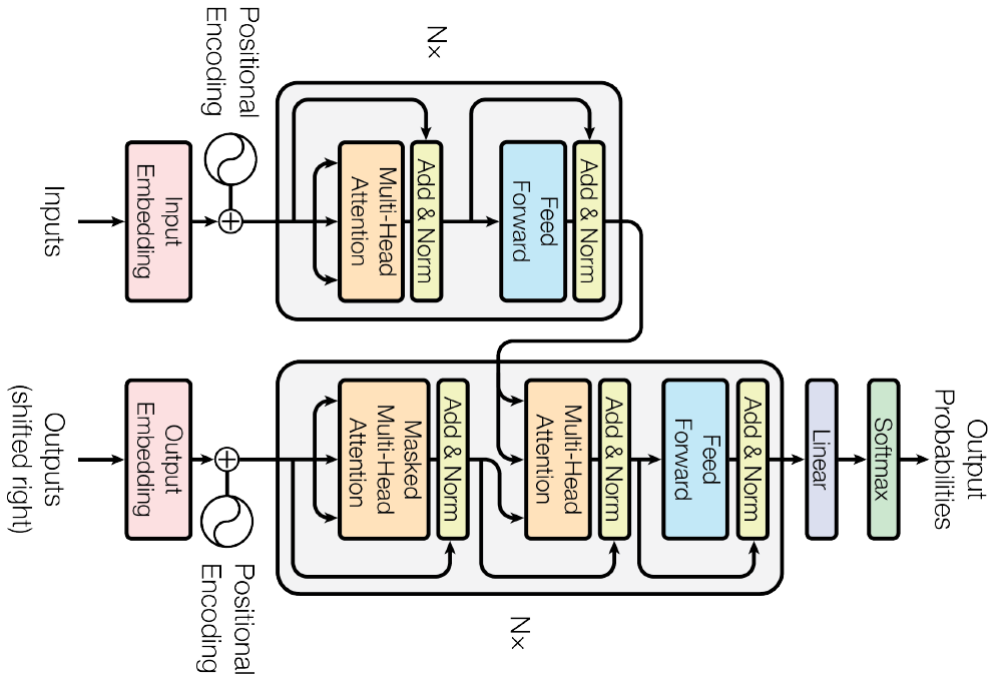


Figure 4: Transformer Architecture

Source: (Vaswani et al., 2017)

Several companies are taking lead and exploiting this architecture to build state of art model for NLP tasks. Below is the list of some selected transformer-based models by various companies.

1. GPT from OpenAI by in their paper "Improving Language Understanding by Generative Pre-Training"
2. BERT from Google by (Devlin et al., 2019) in their paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"
3. XLNet from Google & CMU by (Yang et al., 2019) in their work "XLNet: Generalized Autoregressive Pretraining for Language Understanding"
4. ALBERT from Google Research by (Lan et al., 2019) in their work "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations "
5. T5 (from Google) by (Raffel et al., 2019) in their work "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer"

6. ELECTRA from Google Research & Stanford University by (Clark et al., 2020) in their work "ELECTRA: Pre-training text encoders as discriminators rather than generators"
7. RoBERTa from Facebook by (Liu et al., 2019b) in their work "Robustly Optimized BERT Pretraining Approach"
8. DialoGPT from Microsoft Research by (Zhang et al., 2020a) in their work "DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation"
9. DistilBERT from HuggingFace by (Sanh et al., 2019) in their work "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter".

(Potamias et al., 2020) developed novel architecture RCNN-RoBERTa in their work "A transformer-based approach to irony and sarcasm detection". They developed this architecture using an existing transformer RoBERTa. As mentioned above RoBERTa is developed by facebook research for natural language processing tasks. This novel architecture by authors could predict the sarcasm with 85% to 94% accuracy. In this paper authors has compared performance of various kind of transformers like ELMo , USE, NBSVM, FastText, XLnet, BERT base cased, BERT base uncased, RoBERTa base model, UPF, ClaC, DESC to compare the performance of their novel architecture.

2.5.9 Transfer Learning Approaches

Training a new model from scratch is expensive and time-consuming work. Therefore, recent trends of Transfer Learning are picking up. Some companies or universities who have plenty of resources to develop new models using large amount of data develops the model of various size. They release the models, which need lesser resources to run, for the consumption by other researchers, who have lessor resources & time at their disposable. These released models are called pre-trained models. We can use models as is or with some fine tuning, based on our need.

The pre-trained models are developed using corpus of some language, some task and text from some domain. The beauty of these models is we can finetune them using our data for a task which we want to accomplish. This is called transfer learning. Using transformer-based system we can perform three kind of transfers namely *task transfer, language transfer and domain transfer*. In the NLP world task means like classification, comprehension, text generation, next word or sentence prediction etc. When we say task transfer it means a model which is created for let us say classification task can be finetuned for next word prediction or any other task. To use the model, we need to convert the text into vectors using embedding provided by the

transformer. Recently we see a surge of models of various size in the NLP marketplace. Researcher community is happily adopting those for their experiments and getting good results compare to other approaches and techniques mentioned earlier.

2.6 Approaches to Handle Key Challenges in Sarcasm Detection

2.6.1 Handling Figurative Languages

Figurative language is the language used by intellectuals or those who have a good command over language. If you take a literal meaning of a sentence written in figurative language you will not get anything useful and meaningful. Many times educated people of the society want to communicate some idea or message but they use simile or old proverbs or chose words which are not directly related to the situation but the gist of that incident or proverb has parallel to the situation in hand. Figurative language is work of intellectual caliber and many times it is not easy even for human to understand the message. For example "My daughter is apple of my eyes" मेरी बेटी मेरे आंख का तारा है" If you miss the presence of figurative language in this sentence then you will miss the meaning of this sentence.

(Potamias et al., 2020) in their work "A transformer-based approach to irony and sarcasm detection" claims their novel architecture RCNN-RoBERTa performs well on the figurative language. (Nozza et al., 2016) in their work "Unsupervised Irony Detection: A Probabilistic Model with Word Embeddings" claims that if we integrate probabilistic models like TIM with word embedding then we get promising results in detecting irony and sarcasm.

2.6.2 Handling Limited Data in Sarcasm

Although we did not find research work which tells how much percentage of our day to day communication is sarcastic, but we know from our day to day communication that percentage is very less. You just observe yourself or family members around for one day and count how many times you used sarcastic language. Due to this reason, we do not have enough good size dataset of sarcastic communication. Hindi & Hinglish being one of the least NLP resource languages has too little data to build a good sarcasm detection system.

Researchers takes either of the two approaches to handle imbalance dataset. In first approaches they do not take more non-sarcastic sentences than they have sarcastic ones, this is under sampling of non-sarcastic sentences. In second approach they do over sampling of sarcastic sentences. But it is extremely complicated to create sarcastic sentences with raw data. It is easy to get non-sarcastic sentences but to build our dataset we will not go for collecting more than 1000 non-sarcastic sentences, because we are planning to have 1000 sarcastic sentences in our dataset.

In either of the cases if dataset size is small for the training purpose, we use cross validation techniques. In this technique we create multiple fold of the same dataset using random sampling and then use the fold for the training purpose. Let us say our data set has only 1000 records and it is balanced dataset. If we create a 5 folds cross validation for the training purpose then 5 folds of 200 records will be created from these 1000 records. These 5 folds should have same distribution of the classes. Every time we create new folds there will be different set of records in those folds. After 5 folds are created, we can use 4 folds for training and 1-fold for validation purpose. Thus, we run train our model 5 times and every folds gets opportunity to become validation set. If our experiments, we will use 5 fold cross validation to know the best parameters.

2.6.3 Handling Out of Vocabulary Issues (OOV)

To create word embedding vector which represents all the possible words and their possible usage in different context we need a huge corpus. Not only this, if we have huge corpus of political news or short moral stories that will not represent the same words which are used in the context of medical, physics, philosophy, finance etc. For example, “Interest of various stakeholders is increasing in the recent peace talk process”. This is a statement from normal news. But “Banks are continuously increasing interest and it is making capital more costly” is a statement from financial news. Same word “interest” in financial news has different context than when it is used in normal life. To make sure that final word embedding represent all the possible context we need to include corpus of all the possible domain’s data. But this is difficult task as of today. Because of limited good quality corpus from all the domain of business, science, technology, culture etc.

Due to this reason, at training or prediction time, when we are looking for a word vector for a new context and if word embedding is not available then that word becomes OOV word. When

our dataset has many OOV words then training task will not be able to generate a model which can perform NLP, NLU task with good results. Similarly, if word is available at the time of training but it is not available at the time of validation or in real environment then due to OOV NLP, NLU task performance will be poor, and nobody will use that model.

OOV problems becomes serious when we are using a dataset for training which has words from multiple languages and multiple scripts are used to write those words. This is the typical case of Hinglish language especially in social media or whatsapp communication between Indians. Although there is no silver bullet solution for this OOV problem but if do following we can address this problem to a large extend.

A- Use large corpus
B- Use corpus of different domains
C- Use corpus which has text written in multiple scripts
D- Use corpus which has words from multiple languages
E- Instead of creating context-based vector for words, create subwords from the word and create context vector of those. This is the approach used by fastText of Facebook.

2.7 Embedding

Computers cannot understand text so we need to convert them into numbers. But how to convert a word, phrase, sentence, dialogue, paragraph, chapter, news article, book or encyclopaedia in number? Broadly there are two approaches one is frequency based and another is prediction based.

TF-IDF: Term frequency inverse document frequency is frequency based embedding approach. This is a numerical statistic technique that is intended to reflect how important a word is in a collection or document. TF-IDF numbers of a word imply a strong relationship with the document they appear in, it suggests that if that word were to appear in a query, the document could be of interest to the user, (Ramos, 2003) .

CBOW: Continuous bag of words is a prediction-based technique. It predicts the probability of the word if a context is given. Context window is number of words around the word. Context window of size one means one word left and one word right of the main word. (Wang et al., 2017)

Skip-gram: Skip gram is another prediction-based technique. If we want 3 gram one skip, skip-gram from a sentence "I hit the tennis ball" then we get following skip-grams "I hit the", "hit the tennis", "the tennis ball". This gives us good context understanding. However with this approach a problem of sparsity of the word becomes more severe, (Van Brunt, 1987).

2.7.1 Absolute Embedding

Word embedding like TF-IFD are absolute word embedding approaches. In these approaches word meaning is fixed irrespective of the context a word is used. We know from our experience that meaning of same word can change from one domain to another and one context to other. For example, "गया गया गया". English meaning "Gaya went to Gaya". First word is subject, second word is a verb, and the third word is a location. Absolute embedding approaches cannot handle this kind of text and because of wrong vector the classification task will be incorrect.

2.7.2 Contextual Embedding using full word

Three popular and most used absolute embedding vectors are **glove**, **word2vec** and **freebase**. **Glove840B** is pretrained word vector with 940 billion tokens. This is developed by Stanford university. **Word2vec** [GoogleNews-vectors-negative300.bin.gz] is pretrained word vector with 100 million tokens. **Freebase** [freebase-vectors-skipgram1000.bin.gz] is pretrained word vector with 1.4 million tokens. Word2vec and Freebase are developed by google using google news dataset. In the contextual embedding different meaning of one word in different context can be represented by different vector of the same word. Contextual embedding is done using skip-gram and CBOW. Full word is used to develop this kind of embedding. Issue with this kind of embedding is OOV. If you create word vector using this embedding post lemmatization of word then context is not fully represented but OOV problem will be less. If you develop word vector using this embedding without lemmatization, then OOV problem will be more and matrix will be too sparse.

2.7.3 Contextual Embedding using subwords

As discussed above contextual Embedding using full word cause OOV problem during the training. To address that problem this technique, create subwords from a word and then create word vector of those subwords. Final word vector is sum of all these vectors. **fastText** uses this technique to create word vector. Fasttext treats each word as composed of character ngrams. So the vector for a word is made of the sum of this character n grams. Let's say there is a word "apple" in the sentence so to get the word vector of "apple" we need to sum all vectors of the

n-grams of apple “<ap”, “app”, ”appl”, ”apple”, ”apple>”, “ppl”, “pple”, ”pple>”, “ple”, ”ple>”, ”le>”. Assuming ngram-min is 3 and ngram-max is 6. This embedding technique also uses n-gram and CBOW for creating word vector.

In their paper, “Adaptive GloVe and FastText Model for Hindi Word Embeddings”, (Gaikwad and Haribhakta, 2020) states that AGM gives better results than GloVe and FastTextWeb. They also mentioned that FastText embeddings which are trained on FastTextHin (Hindi Monolingual corpus) produce better results than FastTextWeb. Google research has introduced a multilingual BERT which is capable of working with more than 100 languages (Romano, 2020).

2.8 Types of Sarcasm Detection System

Sarcasm detection systems can be classified in following ways

- **Architecture Used**: Based on the architecture used to develop the system.
    - Rule Based
    - Classical Machine Learning Based
    - Neural Network Based
    - Transformer Based
- **Domain Specific**: Based on the domain it serves.
    - Health
    - Education
    - Travel
    - Social
    - Generic (It is extremely difficult to build a generic SDS)
- **Mode of Communication Based:** This classification is based on the mode of inputs it can accept to perform the classification.
    - Text Based Systems: They can process only online or offline text is used as an input.
    - Voice Based Systems: They can process only voice signals.
    - Video Based Systems: They can process only videos.
    - Image Based Systems: They can accept only images.

- Multimodal System: These systems can take any form of input to perform the classification. It is challenging task to build a SDS which can take all type of inputs, as mentioned above.

- **Time of Detection Based**
  - Realtime Systems: In real time message can be classified as sarcasm or not. For example, as soon as message is delivered on whatsapp, twitter, facebook receiver get a different kind of tick message that it is sarcastic message.
  - Batch Systems: At the end of day or any other frequency, based on the need, all the messages or text can be processed in batch to know how many of them were sarcastic.
- **Language Script Based:** This classification is based upon spoken Language used to write message and written script used to write message.
  - Language Specific: Only for specific language like German or Japanese or Hindi etc.
  - Multiple Language: Can support any spoken language of the world. It is very challenging work to develop such a model.
  - Script Specific: Only for a specific script like Roman or Devanagari or Chinese etc.
  - Multiple Scripts: Can support any script of the world. It is incredibly challenging work to develop such a model which supports all the scripts of the world

2.9 Summary

Thus, we see many researchers have tried to perform the task of sarcasm detection and achieve different accuracy or F1 score depending upon their experiment setups. They have tried different feature extraction techniques and applied those features on different classification algorithms. Most of the work has happened in English language and results are not consistent because results varies due to quality of text in dataset, domain, classification techniques used, features used, data source used etc. Some work has been done for Hindi language and other Indian languages. We did not find any work in Hinglish language which is beyond Twitter dataset. If we observe the table in Appendix B we cannot say with certainty that there is any significant improvement in sarcasm detection results when we use transformers or CNN. We want to experiment with different features and classification algorithms and understand what best results we can achieve when want to detect sarcasm in Hinglish language text.

CHAPTER 3 : RESEARCH METHODOLOGY

In this section we are going to discuss a high-level approach to accomplish the research goal. The flow of discussion in this section is as following 3.1. Dataset, 3.2. Feature Engineering, 3.3. Overview of Our Approach, 3.4. Model Building, 3.5. Evaluation Metrics & Reporting, 3.6. Development Tools.

3.1 Dataset

3.1.1 About Dataset

We started building dataset using Hindi tweet dataset.[17] This excel file had total 442 records. But this sheet does not have labelled data. We cleaned this file, removed ambiguous sentences and put data in our required format. For our project we needed data in the two-column format 1- Sentences 2- Label. After cleaning this data, we had 300 labelled sentences, but this is not sufficient for building a reliable sarcasm detection model for any language. So, we decided to expend this dataset to 2000 sentences with balance data, i.e. 1000 sarcastic sentences and 1000 non-sarcastic sentences. This new dataset has data from tweet as well from normal text or story blogs. All the sources we used to scrap the Hinglish data are available in github file.

3.1.2 Data Sourcing

Sarcasm data in Hindi and Hinglish language on internet is very less. Whatever data is available it is too scattered and painful to extract the data to build a reasonably good size dataset for model building. After lot of surfing on internet we finalized 36 twitter accounts, 22 blogs, and 2 hashtags to scrap the data. To extract the tweets from 36 twitter accounts we wrote a python code using tweepy api. To extract the tweets from 2 hashtags we did little change in the earlier code and could scrap the tweets. To extract sarcasm from blog post we followed two steps. 1- Copied text from the blog post. 2- Read the blog text and break the sentence wherever it looks logical. All the tweets and sentences are put together in one csv file. All the text put in one column “Sentence”. Sentence id is generated for each sentence

3.1.3 Dataset Cleaning

Most of the text from blog was clean but twitter had uncleaned, unstructured sentences. We know that tweet text is unclean because it has text from different languages, in different scripts, extra space, emoticons, non-text sign like "~" ":", "<" etc, flag sign, line break, over used words

[17] https://github.com/rkp768/hindi-pos-tagger/tree/master/News%20and%20tweets (Accessed on 26-Jun-20)

like ".....", "??????", "beau.....tiful", "!!!!!!". Blog text may also have this kind of text but chances of that is extremely less. Now onwards we will not refer this as tweet or blog text but as sentences. Save all the clean sentences text in a new csv file. We wrote a python script to clean all the text. We used following checklist to clean the text, this file is available in github.

3.1.4 Sentence Labelling

Because of various reasons as mentioned earlier, a sentence cannot be labelled as sarcastic by all the people. People have different opinions, and it varies based on individual's personality, education, environment, mood at a specific time and other human personality factors. Before we proceed with our dataset, we wanted a dataset which has unbiased labels. Our dataset has 2368 sentences. To label these sentences as sarcasm we used three annotators who are native speakers and use Hinglish in their day to day communication. All three annotators labelled each sentence independently. Whatever was the max vote for a sentence that label was finally assigned to the sentence.

3.1.5 Dataset Structure

1. Dataset will have 4 columns "Id", "Sentence", "Label", "Twitter"
2. Sentence: Sentence is text of the tweet or any normal sentence.
3. Label: This column will have 0 for normal sentence and 1 for sarcastic sentence.
4. Twitter: This columns will have "Y" if sentence is from twitter else "N"

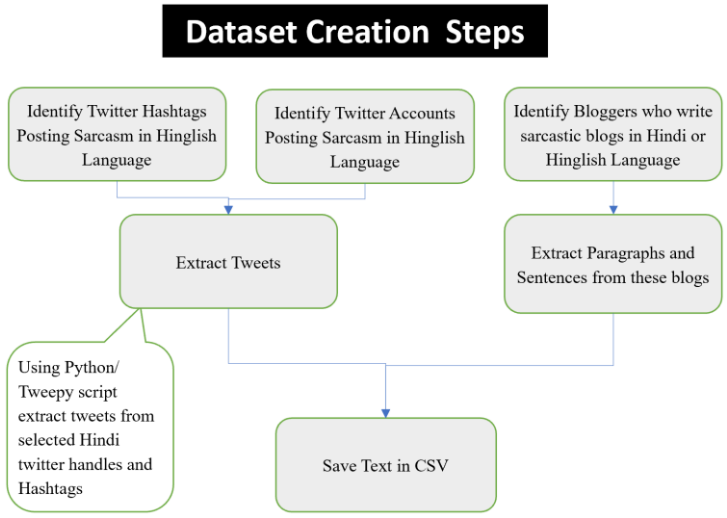


Figure 5: Steps to Creating Dataset

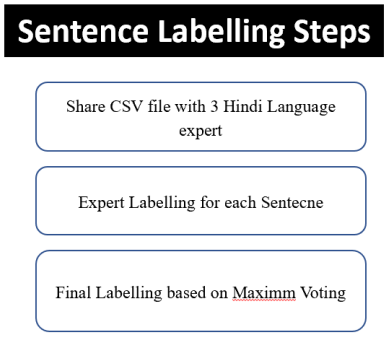


Figure 6: Steps for Labelling Sentences

3.1.6 Feature Engineering

Linguistic Features

We will explore following features.

a) POS based
b) Hashtag
c) Emoticon
d) English Language words

Word Embedding

We will explore following word embeddings.

1. TF-IDF
2. fastText
3. BERT

3.1.7 Embedding and Linguistic Feature Dataset

Following 10 embeddings will be created using our clean dataset. The "combined feature" dataset will be created with linguistic features and the best embedding.

A. Linguistic Features
    1. A dataset only with linguistic features
B. Non-Transfer Embedding
    2. TF-IDF Embedding
    3. BOW embedding
    4. Word2vec embedding
C. Transfer Embedding: fastText Embedding
    5. IndicFT
    6. Ft300Wiki
    7. fastText (Using fastText Library)
D. Transfer Embedding: BERT Embedding
    8. IndicBERT
    9. mBERT
E. Combined Features
    10. A dataset with linguistic + Best Embedding (depends upon results)

Except CNN & RNN based models, all the models we are building will be build using above datasets. Metrices of these models will be compared to see which model works best on which

type of features. CNN & RNN models will be build using tokens of clean text and best embedding transfer.

3.2 Model Building

3.2.1 Test-Train Split

Because of dataset is very small therefore we will use train-test split of 90:10. 90% sentences will be used for training and 10% of the sentences will be used for validation. We need to make a note, in cross validations test results are based on the folds of 90% of the sentences. Cross validation technique can tell us which set of hyperparameters on which fold gives best test results. A model with best hyperparameters will be validated against 10% validation data.

3.2.2 Handling Small Dataset size

Our dataset has 2000 records. This is a small size dataset. Because dataset is not large enough therefore, we will use cross validation of 5 folds for model building.

3.2.3 Algorithms, Architecture for Modeling

We will use following 14 techniques for building classification models. Embeddings discussed in 3.1.7 will be used with all the Classical models as mentioned below.

A. Classical Models
    1. Logistic Regression (LR)
    2. Light Gradient Boosting Method (LGBM)
    3. Naïve Bayesian (NB)
    4. AdaBoost (ADB)
    5. Support Vector Machine (SVC)
    6. Gradient Boost Classifier (GBC)
    7. Random Forest Classifier (RFC)
    8. XGBoost (XGB)
    9. Decision Tree (DT)
    10. Perceptron

B. Neural Network
    11. CNN
    12. RNN

C. Task Transfer Models
    13. **BERT**: mBERT (transformer/pytorch), IndicBERT (TL)

14. **fastText**: FT300wiki, IndicFT (TL)

3.3 Overview of Our Approach

We are starting this project with almost zero data in our hands. So the first steps is create a good size dataset which can be used for our project. The details are mentioned in section 3.1.2. The dataset created is not fit for modelling, so we need to need to clean this dataset. The details are mentioned in section 3.1.3. Following this we need to manually label each record with the help of annotators. The details are mentioned in section 3.1.4. After following all above steps we have clean data in place. At this stage will split our dataset into train test, discussed in 3.2.1. For all the embedding, used for model building, we will ensure that same ID are used for train and test. This is done to compare the results across different classifiers and different embeddings used.

Following that we combine different features (3.1.7) with different classifier (3.2.3) and develop different models. After building all the possible models we will evaluate classifier and embedding. Finally, we take the best embedding & best classifier and combine with lexical features and create a combined model. This will help us understanding, whether we can get any better results than we are getting from best embedding and best classifier.

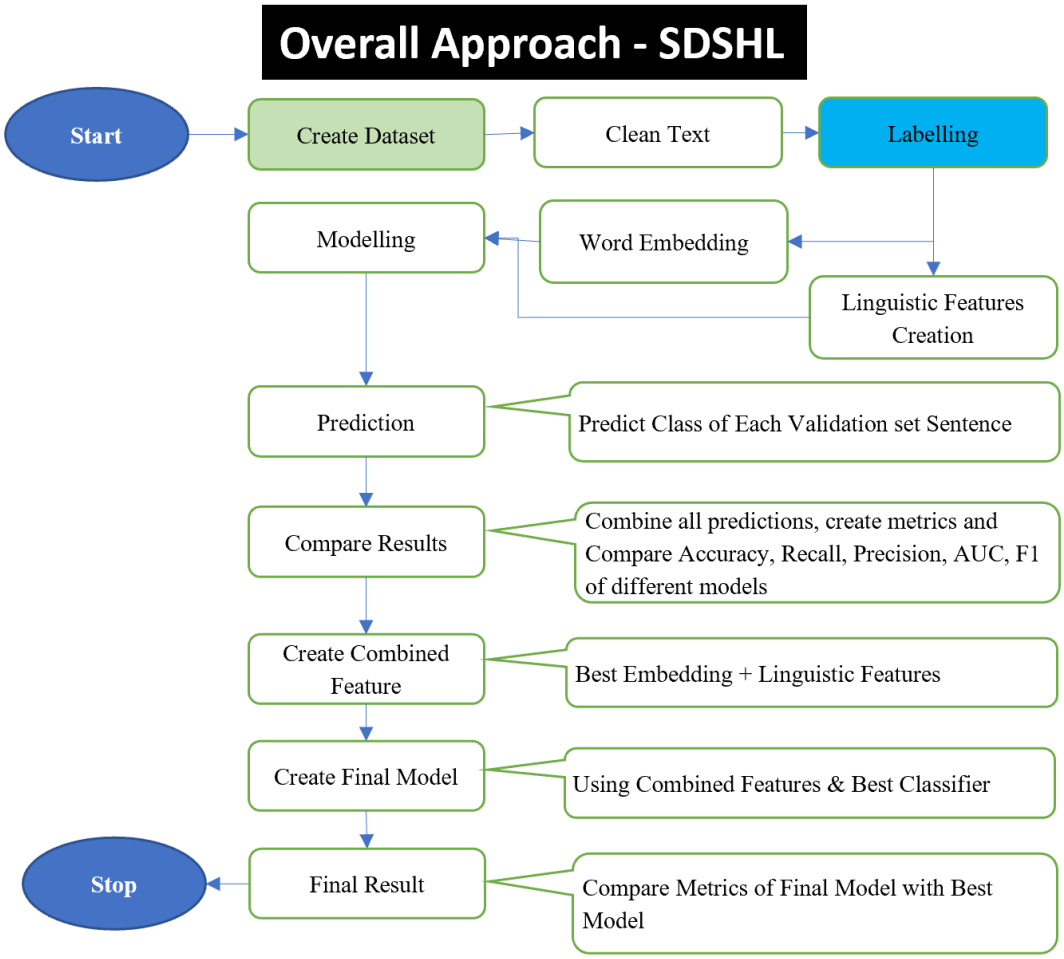


Figure 7: Overall Approach

Classifiers	**Word Embedding**	**Feature Engineering**
1. Logistic Regression (LR)	**No Transfer**	1. Lexical Feature
2. Light Gradient Boosting Method (LGBM)	1. TFIDF	2. Combined = IndicFT + LexicalFeature
3. Naïve Bayesian (NB)	2. Word2Vec	
4. AdaBoost (ADB)	3. BOW	**Task Transfer**
5. Support Vector Machine (SVC)	4. fastText	1. mBERT (Pyrotch)
6. Gradient Boost Classifier (GBC)	**Transfer Embedding**	2. mBERT (Transformer)
7. Random Forest Classifier (RFC)	1. IndicBERT	3. IndicBERT
8. XGBoost (XGB)	2. Multilingual BERT	4. IndicFT
9. Decision Tree (DT)	3. fastText Wiki	5. fastTextWiki
10. Perceptron	4. fastText Indicnlp/ IndicFT	

Table 2: Classifiers, Embedding/Features, Task Transfer Learning Used

3.4 Evaluation Metrics & Reporting

ROC graphs are useful tool for visualizing and evaluating classifiers. ROC are able to provide a richer measure of performance than accuracy or error rate (Fawcett, 2004). From the Appendix B we can notice that most of the researchers either used Accuracy or F1 score to

measure the performance of the sarcasm detection system or sentiment analysis. However, we will use Accuracy, Recall, Precision, F1 & ROC, because they have their relevance depending upon the domain where we use this for sarcasm detection. To understand it better, let us see a sarcasm from hospital, health domain.

A patient says “Hospital administration thinks that I come to hospital because I have lot of money and they have beautiful nurses to chat with” (writing sarcasm in English to make sure more readers understand the impact of choice of evaluation metrics).

Healthcare domain, hospital administrators would like to take a sarcasm seriously and they do not want any sarcasm to be misclassified and they are ready for more False-True (which our system identify sarcastic but in reality they are not). To illustrate the choice of metrics, let’s assume there are 1000 sentences in the real time dataset, 150 are sarcasm and 850 are normal sentences. Let us say Model1 predicts 110 are sarcasm and 890 normal and Model2 predicts 140 sarcasm and 860 normal sentences. Let’s say accuracy of both the models is 90%. If we select Recall and F1 score, then Model1 is better. If we select precision, then Model2 is better. If we need to detect sarcasm in comment box of YouTube channel of some political party, then we can go for Model1 which is giving recall of 73%. If we are dealing with some more serious product or service like healthcare, airlines service then we can go for Model2 which is giving Precision score of 63%.

Model1

		Observation		
		FALSE	TRUE	Total
Actual	FALSE	820	30	850
	TRUE	70	80	150
	Total	890	110	1000

Accuracy	0.90
Recall	0.73
Precesion	0.53
F1 Score	0.81
Error Rate	0.10

Model2

		Observation		
		FALSE	TRUE	Total
Actual	FALSE	805	45	850
	TRUE	55	95	150
	Total	860	140	1000

Accuracy	0.90
Recall	0.68
Precesion	0.63
F1 Score	0.77
Error Rate	0.10

Table 3: Performance Metrics Selection

The result of prediction will be compared using AUC, F1, Accuracy, Recall, Precision. Our dataset is balanced dataset therefore even Accuracy is good enough measure to compare the performance of models.

3.5 Development Tools

Language: Python 3.0>

ML Libraries

- Matplotlib
- Seaborn
- Pandas
- Numpy
- Sklearn
- RE

Indian Language Libraries

- NLTK
- iNLTK

Classical Modelling

- Logistic Regression
- Light Gradient Boosting Method
- Naïve Bayesian
- Adaboost
- Support Vector Machine
- Gradient Boost Classifier
- Decision Tree
- Random Forest Classifier
- Decision Tree
- Perceptron

Word Embedding

- TF-IFD
- BOW
- Word2Vec
- fastText (Subword Based)
- indicFT (Subword Based)
- FTwiki_hi300 ((Subword Based)
- indicBERT (Transformer Based)
- mBERT (Transformer Based)

Neural Network

- CNN
- RNN

Framework

- PyTorch
- Transformer
- Tensorflow
- Keras

3.6 Summary

We will develop a dataset of 2000+ sentences. Some text will be taken from twitter and some other will be taken from Hindi blogs. Data will be cleaned and labelled with the help of native speakers. For creating features of the dataset, we will extract linguistic features from the sentences. We will also use word embedding like TF-IDF, word2vec, fastText, BERT to create features. For developing models, we will use classical machine learning models like LR, LGBM, NB, ADV, SVC, GBC, RFC, XGB, DT and Perceptron. We will also explore CNN, RNN, fastText and transformers like BERT. For measuring model performance, we will use 5 metrics Accuracy, Recall, Precision, F1 and AUC. With all these experiments we will present our finding which type of features, which word embedding, which classifier gives best results for Hinglish Sarcasm classification

CHAPTER 4 : ANALYSIS

4.1 Introduction

Our dataset has two types of text. Twitter text and regular blog text. This was done to understand that how can we make a system which can predict sarcasm on Hinglish text without bothering source of text. Our system should be able to work for both kind of text mixed text of twitter and clean text of a blog. Our dataset has two classes namely Sarcasm and Non-Sarcasm. Our dataset has 2000 sentences, 1000 sentences are sarcastic sentences and 1000 non sarcastic.

	Non-Sarcastic (50%)	**Sarcastic (50%)**	**Total**
Blog Text	179 (39%)	283 (61%)	**462 (23%)**
Twitter Text	821 (53%)	717 (47%)	**1538 (73%)**
Total	**1000**	**1000**	**2000**

Table 4: Class Distribution in Dataset

In total our dataset is balanced but distribution of sentences for two different type of text is not balanced. On the other hand, distribution of classes for two type text of dataset is not same. However, our focus was not on understanding which class of the text can be classified better so we ignored this disbalancing in our experiments. We are focused on building a system which can predict whether a input sentence is sarcastic or not.

Our dataset has 4 fields ID, Sentence, Label(1- Sarcastic, 0-Non-Sarcastic), Twitter (Y- Yes, N- No). We performed train test split on our dataset. Train has 90% of the sentence and test has 10% of the sentences. For all the embedding and all the experiments, we ensured that the ID of train and test set is same across all the experiments. This helped us tracking the predicted classes of sentence for different model.

There are **four** ways of creating classification models. **1-** Task Transfer, **2- N**eural network**, 3-** using Lexical Feature (manual feature engineering), **4-** Embedding.

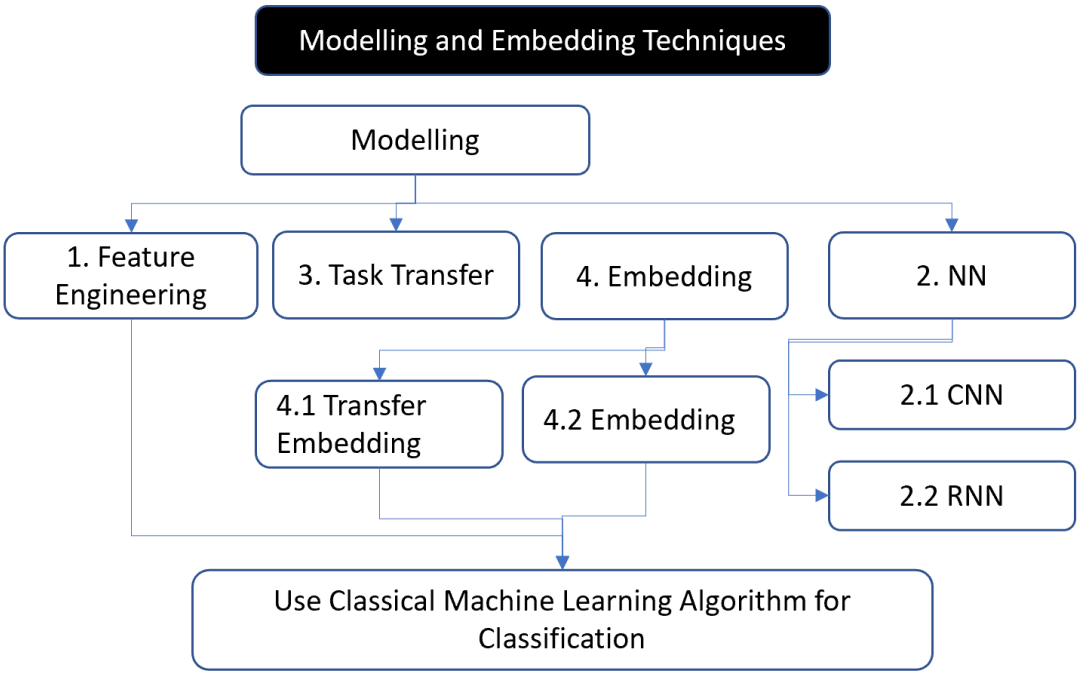


Figure 8: Modelling & Embedding Techniques

4.1.1 Task Transfer

In the **Task Transfer** models we transferred classification task from pretrained model to our model. For this we downloaded the pretrained models and finetuned the downloaded classification model using our trained dataset. Testing was performed using test dataset, which was decided earlier. In this experiment we did not do any embedding explicitly. We provided tokenized text as an input and all work is done by the pretrained models. We used 5 models for task transfer.

1. mBERT / BERT Multilingual (by Google) using Pytorch Implementation. Multilingual BERT is discussed by (Pires et al., 2020) in their work "How multilingual is multilingual BERT?". BERT is developed by (Devlin et al., 2019)
2. mBERT [18]/ BERT Multilingual (by Google) using Transformer Implementation
3. IndicBERT [19](by AI4Bharat) (Kakwani et al., 2020)
4. IndicFT [20](by AI4Bharat). (Kakwani et al., 2020)
5. fastText Wiki[21] (by Facebook) (Bojanowski et al., 2017)

[18] https://huggingface.co/bert-base-multilingual-uncased
[19] https://indicnlp.ai4bharat.org/indic-bert/
[20] https://indicnlp.ai4bharat.org/indicft/
[21] https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.hi.300.bin.gz , https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.hi.300.vec.gz

4.1.2 Neural Network

When we are using neural network for developing models, we create CNN & RNN architectures. In our experiment we created tokenizer using Kera's tokenizer and train data. We trained our CNN & RNN model using these tokens of train dataset. Tokens for the test data were created using the same Keras tokenizer. Our model is tested using this tokenized test dataset. We selected two pretrained embedding (indicFT & fastTextWiki) and transferred the embedding to CNN and created two CNN models. Next, we will discuss third and froth technique of modelling.

4.1.3 Lexical Features Method

Creating lexical feature is a manual work. Using feature engineering techniques for text we created following features.

1. Number of NOUN,
2. Number of ADP,
3. Number of VERB,
4. Number of AUX,
5. Number of PRON,
6. Number of PROPN,
7. Number of PART,
8. Number of DET,
9. Number of PUNCT,
10. Number of ADJ,
11. Number of SCONJ,
12. Number of CCONJ,
13. Number of NUM,
14. Number of ADV,
15. Number of INTJ,
16. Number of X (OOV words),
17. words (total number of words in the sentence),
18. Eng_words (number of English words),
19. No_Emotion (number of emoticons),
20. No_Hashtag (number of hashtags).

Train and test datasets have same sentence as we fixed earlier. With these lexical features we developed various models using classical ML algorithms. For POS features we used stanfordnlp[22] hindi library. This library is developed by (Qi et al., 2018).

4.1.4 Embedding

We know text cannot be used by any machine learning algorithm until it is converted into some numeric equivalent. Broadly there are two ways for this. First is Lexical Feature engineering and second is embedding. There are many tools, techniques, frameworks, and models for embedding. After the text has been converted into numeric equivalent the output is called embedding.

4.1.4.1 How Embedding works?

Primarily there are two kinds of embedding but in principle there can be many. Two embeddings are word embedding and sentence embedding. Sentence embedding can be generated from the embedding of the words used in the sentence. The most popular way of generating sentence embedding is average the embedding of all the words used in the sentence. For example, there is sentence "I love chocolates" which has 3 words. To understand it further let us say we have 5-dimensional embedding of these 3 words as following

I = [.4534 .8345 .1475 .3495 .2309]
Love = [.4567 .1293 .8734 .5643 .0934]
Chocolate = [.0987 .0324 .9345 .1234 .3456]
Then sentence embedding of this sentence using average of the embedding of three words would be pairwise sum and divided by three
I Love Chocolate = [.3363 .3321 .6518 .3457 .2233]

Thus, word is a lowest level of embedding and embedding of phrase, sentence, paragraph, chapter etc can be generated by following above technique. But we need to understand, if we use word as a lowest level embedding then many times a new word can appear in the input text at testing time or during production run and this word was not available during the training. This will cause out of vocabulary problem and hence no embedding would be found. Further, if word embedding is not available then this word will be ignored during sentence embedding.

[22] https://github.com/stanfordnlp/stanfordnlp

To avoid this problem there are two broad approaches. First is use subwords to get embedding of word. Second is have huge size corpus which has all the words and their usage in all the contexts. Both the approaches have their advantages and disadvantages. Both the approaches have been used by the researchers to develop word embedding. Embeddings of the same word can vary and it depends upon following factors

1. The algorithm used,
2. Domain of text corpus used (finance, medical, political news etc),
3. Culture from where text was used (American, European, Asian etc),
4. Gender (if text represent a specific gender),
5. Age group for which text was written (Children text, adult text, school text, college text),
6. Language of the text corpus (Sanskrit, Hindi, Tamil, Kannada, Telegu, English, Arabic, French etc),
7. Script of the text corpus (Devanagari, Roman, Kannada, Telegu etc),
8. Religion of the people which text corpus is about (Hindu, Buddhist, Islamic, Christian etc),
9. Era when text was written (2000 years old text, 500 years old text, 19th century text, 21st century text.

Research department of organization like Google, Facebook, MIT has developed different embedding using various algorithms. Some of these algorithms are language specific and some are multilingual.

We created ten embedding using our dataset. Some of the embedding are based on pretrained embedding with fine tuning to our train data. This technique is called embedding transfer learning. Some other embeddings are generated using our train data. The corpus and computing power available to big companies like Facebook, Google etc is non comparable to our corpus size and computing power. Due to this reason our embedding did not produce that good results as embedding transfer learning could produce.

Ten embeddings used are.

1. **TFIDF** : Term Frequency Inverse Document Frequency
2. **BOW** : Bag of Words

3. **Word2Vec** : Word2Vec algorithm is developed by **Google** and it uses CBOW and Skip-Gram. We created our embedding using Word2Vec.
4. **IndicBERT** : This is based on ALBERT (a lighter version of BERT). BERT and ABERT both are developed by Google. It is Transformed based model. IndicBERT is created by AI4Bharat using Hindi corpus. This model is created by **AI4Bharat**. We used this IndicBERT for embedding transfer and task transfer.
5. **BERT Multilingual (mBERT)**: This is created by **Google** and it supports Multiple (104) Language. This is transformer based model. We used this model for embedding and task transfer. We implemented mBERT using transformer to create embedding. To do the task transfer we used mBERT with transformer and pytorch and got two different predictions.
6. **fasttext** : fastText library is developed by **Facebook**. We used this to create our own embedding. (No transfer learning)
7. **fastText_wiki** : This model is developed by **Facebook** using Hindi wiki corpus. We used this model for embedding transfer and task transfer
8. **IndicFT** : This is based on facebook's fastText. This model is developed by **AI4Bharat** using Hindi corpus. We used this model for embedding transfer and task transfer
9. **Combined** : We created this embedding from the best embedding (fastText_wiki) + Lexical features. In our experiments we found fastText_wiki embedding is giving the best results therefore we combined it with lexical features to know whether we can get even better results.
10. **Lexical**: It is not an embedding but features created using feature engineering techniques.

4.1.4.2 Embedding Method

We created two types of embedding. In the first type of embedding we transferred the embedding of existing pretrained models. For this we downloaded some most relevant and popular pretrained models. We performed fine tuning using our train dataset and created final model for word embedding. Using this final model, we created embedding for our complete dataset. Train and test set of embedded sentence has same ID as in mentioned earlier. Various classical machine learning algorithms were used for developing classification models using this train data. Test was performed on embedded test dataset.

In the second type of embedding we created embedding using our train dataset and no-transfer learning of any kind is used. For this we used algorithms like TFIDF, Word2Vec, BOW and fastText. After models are created on train dataset we use that for creating embedding for entire dataset. ID of train and test dataset remain same as mentioned earlier. We used various classical machine learning classification algorithm for developing various models and test those models on test dataset created from this embedding.

4.2 Architecture, Parameters of Classifier & Embedder

4.2.1 Parameters – Embedding without Transfer Learning

4.2.1.1 TFIDF

Vectorizer: TfidfVectorizer

Parameters: max_features=300, ngram_range=(1,2)

pca = PCA(n_components=200)

Final embedded dataset has has 100 features which represents the input text.

Dimension: 200

4.2.1.2 BOW

Vectorizer : CountVectorizer

Parameters: max_features=300, ngram_range=(1,2)

pca = PCA(n_components=200)

Final embedded dataset has has 100 features which represents the input text.

Dimension: 200

4.2.1.3 Word2VEC

Vectorizer: Word2vec

Parameters: feature_size=15,

window_context=20,

min_count = 1,

sg=1,

sample= 1e-3,

iter=5000

Dimension: 15

4.2.1.4 fastText

Vectorizer: gensim.models.fasttext import FastText

Tokenizer Params:

feature size=50, # Word vector dimensionality

window_context=20, # Context window size

min_word_count = 1 # Minimum word count

skip gram=1, # skip-gram model

sample=1e-3, # Downsample setting for frequent words

iter=5000

4.2.2 Parameters - Embedding with Transfer Learning

4.2.2.1 IndicBERT

Tokenizer: ai4bharat/indic-bert

Tokenizer Params: tokenizer.encode_plus(text,

add_special_tokens=True,

max_length=200) ["input_ids"]

Dimension: 768

4.2.2.2 mBERT

Tokenizer: bert-base-multilingual-uncased

Tokenizer Params: tokenizer.encode_plus(text, add_special_tokens=True,

max_length=200) ["input_ids"]

Dimension: 768

4.2.2.3 IndicFT

Pretrained vector: indicnlp.ft.hi.300.vec

Tokenizer Params : fasttext.train_supervised(train_file,

dim=300, \

lr=0.5,

epoch=25,

wordNgrams=2,

bucket=200000,\

pretrainedVectors)

4.2.2.4 fastText_wiki

Pretrained vector: wiki.hi.300.vec

Tokenizer Params: fasttext.train_supervised(train_file,

dim=300, \

lr=0.5,

epoch=25,

wordNgrams=2,

bucket=200000,\

pretrainedVectors)

4.2.2.5 Lexical

Manual feature engineering. Dimension : 20

4.2.2.6 Combined

This embedding contains features from IndicFT & Lexical.

Dimension: 788

4.2.3 Parameters - Classifiers

Logistic Regression – LR

LogisticRegression (C=.01,max_iter=1000, random_state=100)

Light Gradient Boost Machine – LGBM

lgbm.LGBMClassifier(colsample_bytree=1.0,

importance_type='split', learning_rate=0.1, max_depth=-1,

min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,

n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,

random_state=100, reg_alpha=0.0, reg_lambda=0.0, silent=True,

subsample=1.0, subsample_for_bin=200000, subsample_freq=0)

Naïve Bayesian – NB

Default Parameters

Support Vector Classifier – SVC

Default Parameters

AdaBoost – ADB

Default Parameters

Gradient Boost Machine – GBM

GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
learning_rate=0.1, loss='deviance', max_depth=3,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=100,
n_iter_no_change=None, presort='deprecated',
random_state=100, subsample=1.0, tol=0.0001,
validation_fraction=0.1, verbose=0,
warm_start=False)

Random Forest Classifier – RFC

Default Parameters

Perceptron

Default Parameters

4.2.4 Parameters - Task Transfer

4.2.4.1 mBERT- Transformer

Token: bert-base-multilingual-uncased

Params of Token: pad_sequences(list (map (bert_tokenizer.convert_tokens_to_ids, train_tokens)), maxlen=256, truncating="post", padding="post", dtype="int")

Model: bert-base-multilingual-uncased

Parameter of Model : BertTransformer(bert_tokenizer_multi, bert_model_multi, max_length=200)

4.2.4.2 mBERT – Pytorch

Tokenizer: bert-base-multilingual-uncased

Params of Tokenizer:

list(map(lambda t: ['[CLS]'] + Tokenizer.tokenize(t)[:255] + ['[SEP]'], X))

tokens_ids = pad_sequences(list(map(bert_tokenizer.convert_tokens_to_ids, Tokens)), maxlen=256, truncating="post", padding="post", dtype="int")

Model: bert-base-multilingual-uncased

Params of Model: Batch_Size: 2, Ephoch=10, dropout=10%, add_special_tokens=True, max_length=self.max_length

4.2.4.3 IndicBERT

Tokenizer: ai4bharat/indic-bert

Params of Tokenizer: tokenizer.encode_plus(text, add_special_tokens=True, max_length=200) ["input_ids"]

Model: ai4bharat/indic-bert

4.2.4.4 IndicFT

Pretrained vector: indicnlp.ft.hi.300.vec

Params of ftmodel_indicnlp300_vec = fasttext.train_supervised(train_file,

dim=300, \

lr=0.5,

epoch=25,

wordNgrams=2,

bucket=200000,\

pretrainedVectors)

4.2.4.5 fastText_wiki

Pretrained vector: wiki.hi.300.vec

Params of ftmodel_wiki300_vec = fasttext.train_supervised(train_file,

dim=300, \

lr=0.5,

epoch=25,

```
        wordNgrams=2,
        bucket=200000,\
        pretrainedVectors)
```

4.2.5 Neural Network Architecture

4.2.5.1 CNN Architecture with TL

```
embedding_dim = 200
sent_size =119
batch_size=100
cnnmodel = Sequential()

#embedding layer
cnnmodel.add(layers.Embedding(vocab_size, embedding_dim, input_length=sent_size))

#CNN layer
cnnmodel.add(layers.Conv1D(128, 5, activation='relu'))
cnnmodel.add(layers.GlobalMaxPooling1D())

#FC layer
cnnmodel.add(layers.Dense(10, activation='relu'))
cnnmodel.add(layers.Dense(1, activation='sigmoid'))

#Add loss function, metrics, optimizer
cnnmodel.compile(optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy'])
```

4.2.5.2 CNN with Transfer Learning from fastText_Wiki

```
vocab_size= 9156
Weights = Weight of above vocabulary from from fastText_Wiki finetuned model
Dim=300
input_length = 119
Remaining architecture same as above.
```

4.2.5.3 CNN with Transfer Learning from IndicFT

All parameters remained same as above. Weights are taken from IndicFT finetuned model

4.2.5.4 RNN Architecture

```
embedding_dim = 200
sent_size =119
batch_size=100
rnnmodel=Sequential()

#embedding layer
rnnmodel.add(layers.Embedding(vocab_size, embedding_dim, input_length=sent_size))

#lstm layer
rnnmodel.add(LSTM(128,return_sequences=True,dropout=0.2))

#Global Maxpooling
rnnmodel.add(GlobalMaxPooling1D())

#Dense Layer
rnnmodel.add(Dense(64,activation='relu'))
rnnmodel.add(Dense(1,activation='sigmoid'))

#Add loss function, metrics, optimizer
rnnmodel.compile ( optimizer='adam', loss='binary_crossentropy', metrics=["acc"])

#Adding callbacks
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1,patience=3)
mc=ModelCheckpoint( 'best_model.h5', monitor='val_acc',  mode='max',
save_best_only=True, verbose=1)
```

4.3 Data Visualization

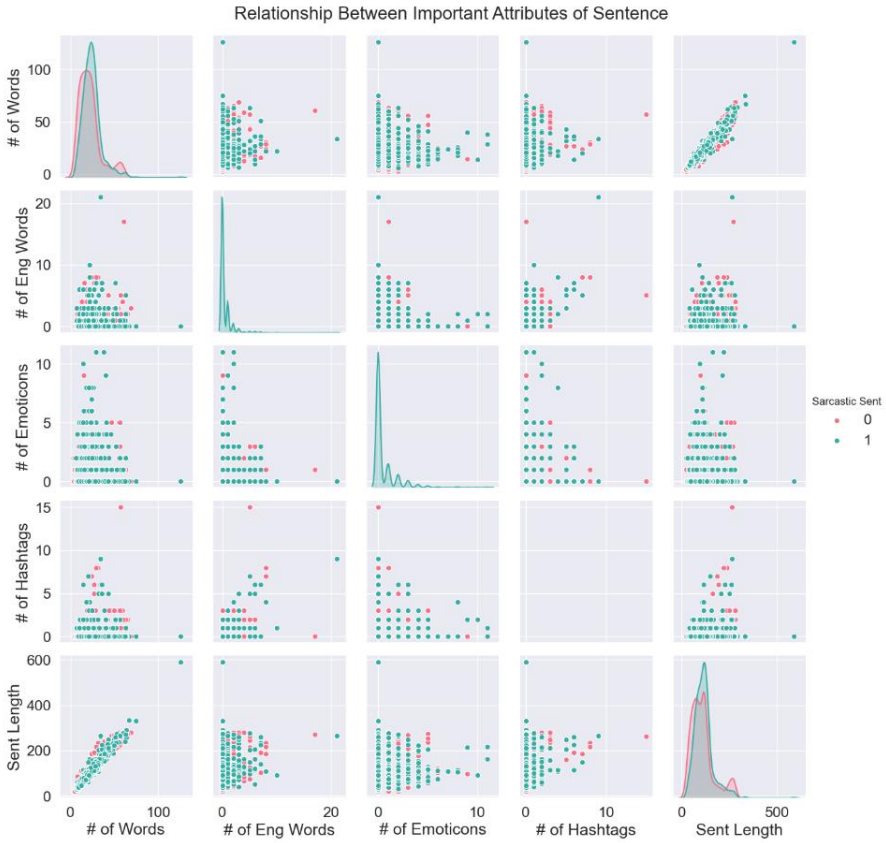


Figure 9: Correlation between different attributes of Sentences - Distribution

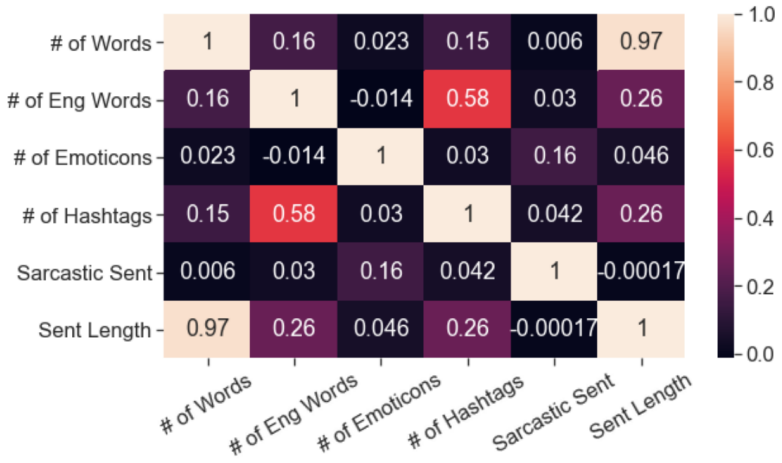


Figure 10: Correlation between different attributes of Sentences - Heatmap

There are following observations from the above table and pair-graphs.

- Longer the sentence more the words (obvious)
- More number of hashtags means more English words used.
- Longer the sentence there will be more English words.
- Longer the sentence more hashtags it will have.

- Rest all relationships are weak.
- Almost no relationship between sentence being sarcastic and number of hashtags used, number of words, sentence length, number of emoticons used.

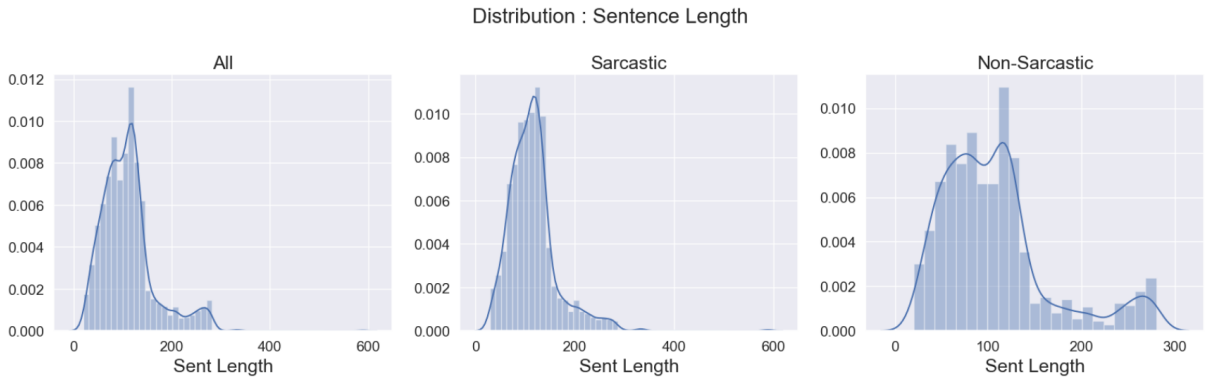


Figure 11: Distribution of Sentence Length of Sarcastic & Non-Sarcastic Sentences

Sarcastic sentences in our dataset has more normal distribution of the sentence length than non-sarcastic sentences.

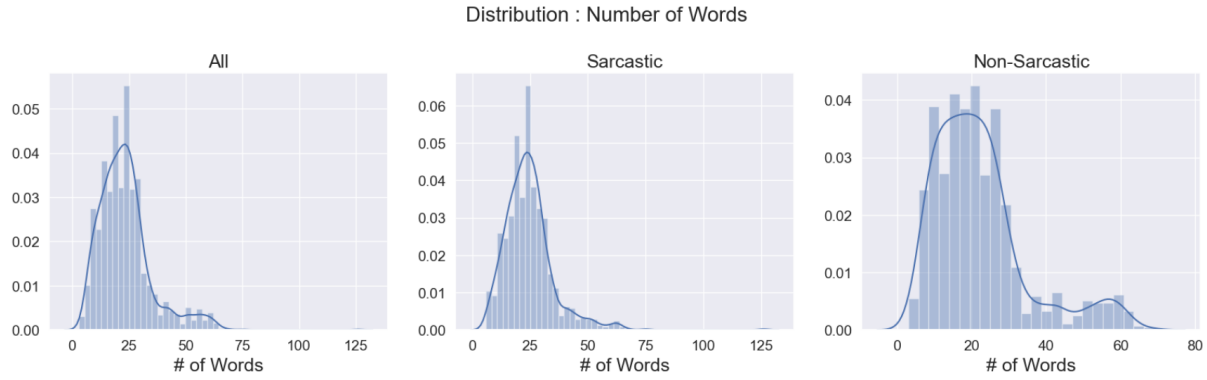


Figure 12: Distribution of # of Words in Sarcastic & Non-Sarcastic Sentences

Sarcastic sentences in our dataset has more normal distribution for “number of words” than non-sarcastic sentences.

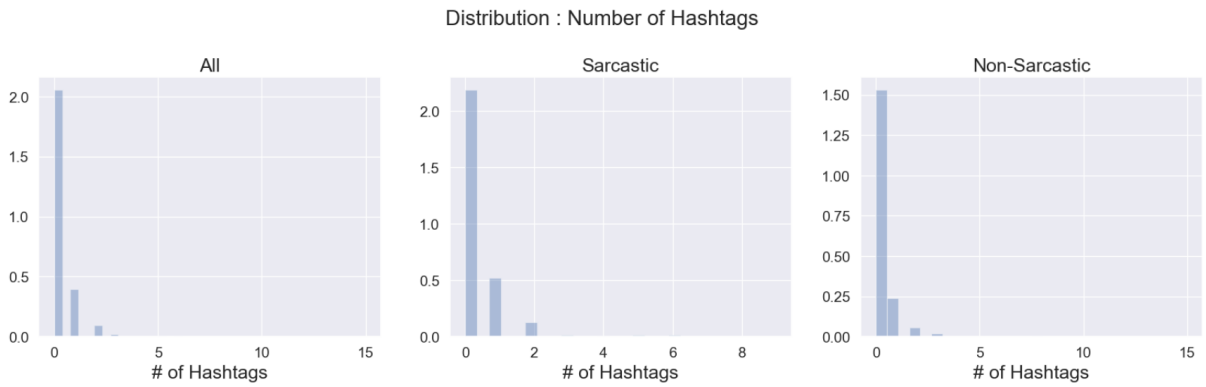


Figure 13: Distribution of # of Hashtags in Sarcastic & Non-Sarcastic Sentences

Number of hashtags used has same distribution for sarcastic and no sarcastic sentences.

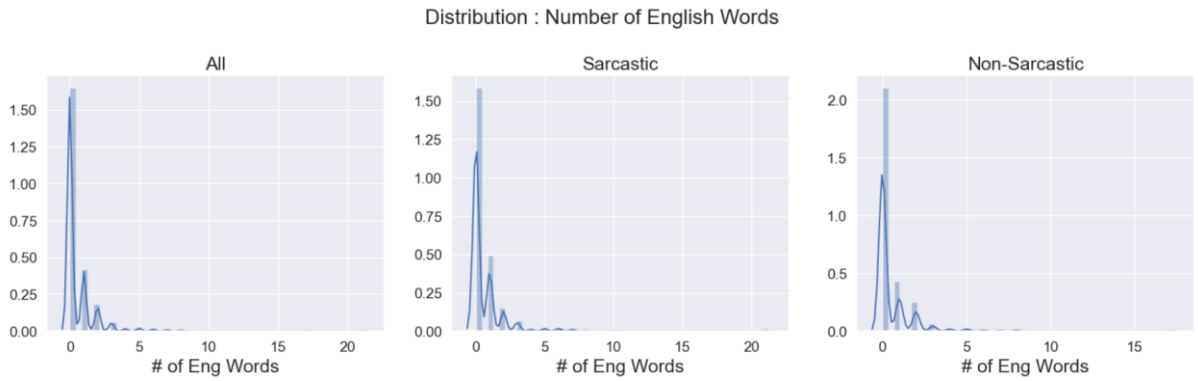


Figure 14: Distribution of # of English Words in Sarcastic & Non-Sarcastic Sentences

Number of English words used has same distribution for sarcastic and no sarcastic sentences.

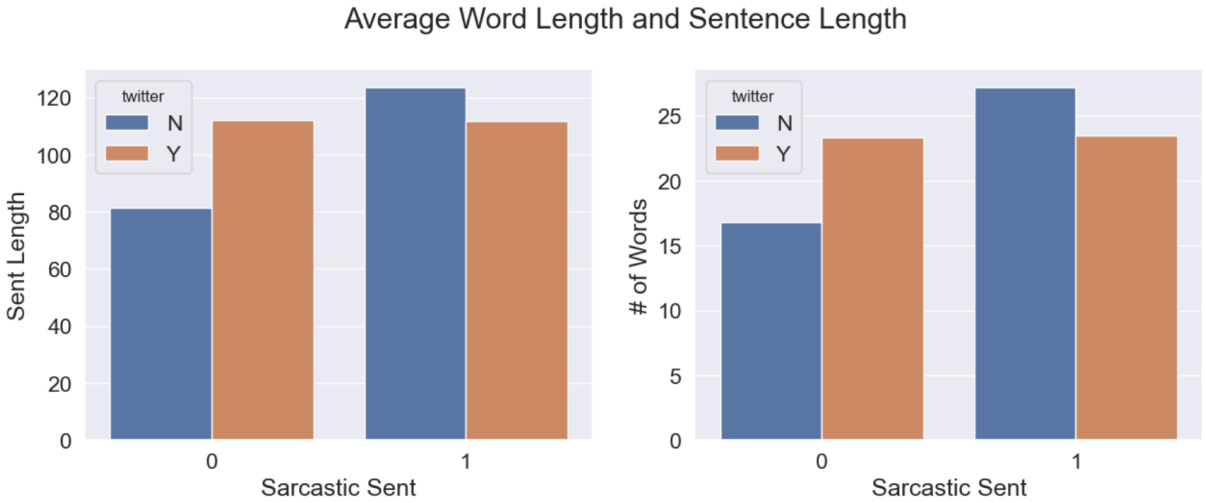


Figure 15: Mean Word Length / Sentence Length of Sarcastic and Non-Sarcastic Sentence

Sentence is sarcastic or not its length remains same in case of twitter, obviously because of twitter character limit. But in normal life people in Hinglish language speak longer to be sarcastic.

4.4 Summary

We created a Hinglish language dataset of 2000 sentences 1538 twitter sentences and 462 text from blogs. 1000 of these sentences are sarcastic and 1000 are non-sarcastic sentences. We performed data visualization to know the characteristics of the data and found our text is good enough for model building. We performed classification task transfer using this data and developed 5 task transfer model. We also did embedding transfer using mBERT, fastTest_Wiki, IndicBERT, IndicFT and created 4 embedding using this technique. Using vectorizers library we created 4 another embedding. We created lexical features and combined the best embedding with lexical features. We created total 100 models using 10 classifier namely LR, LGBM, NB, SVC, ADB, GBM, RFC, XGB, DT & Perceptron. We also created 4 models using CNN and RNN. We used task transfer learning techniques to create another 5 models.

CHAPTER 5 : RESULT AND DISCUSSION

5.1 Introduction

In this project 109 models were developed. Six techniques used and number of models developed are as below.

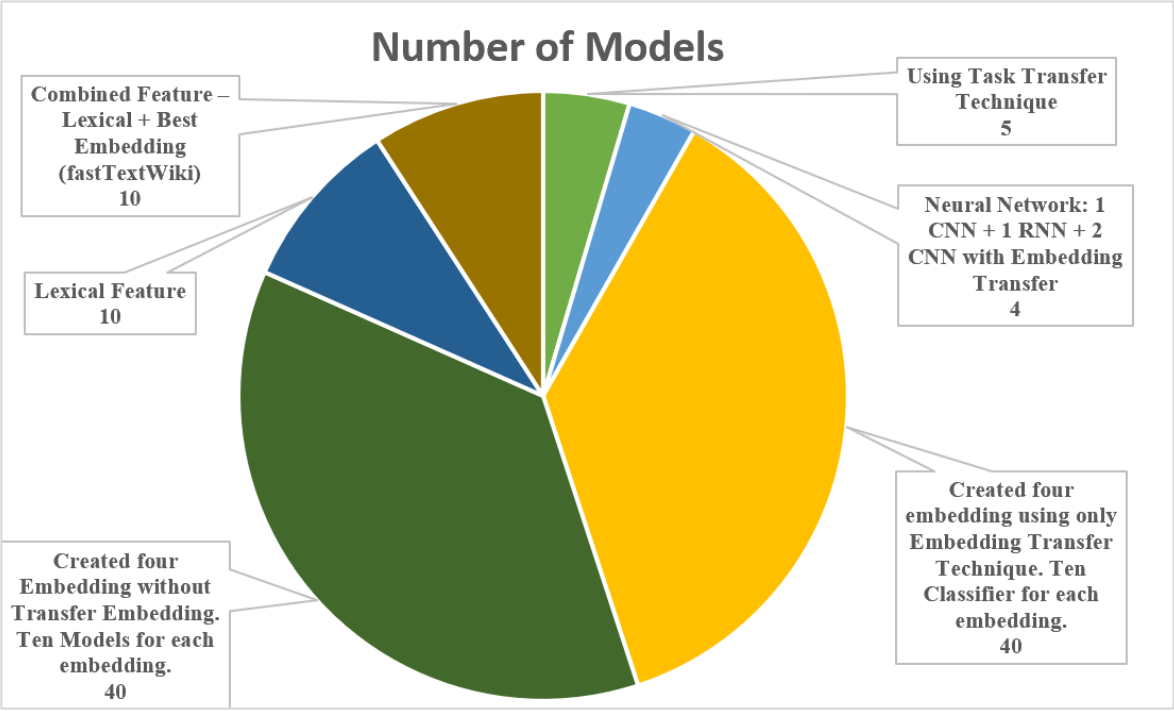


Figure 16: Number of Models Developed

Five metrics are used to measure the performance of these 109 models. Each metrics has its strength and value in decision making. Five metrics used are

- Accuracy: If dataset is balanced and negative and positive classification are equally important. Our dataset is balanced therefore this metrics is good enough to compare the performance of our models. Because other researchers have used other metrics, so we have computed other metrics as well.
- Recall: If goal is to minimise false negative then Recall is used.
- Precision: If goal is to minimise false positive then Precision is used.
- F1 Score: If goal is to minimise both false negative and false positive then F1 is used.
- AUC: All the metrics are influenced by the threshold used to label a class. At different thresholds above metrics of the same model can vary. AUC is the only metric which remains constant for a model irrespective of the threshold used.

5.2 Interpretation & Visualization

Results in all the tables show are in descending order of accuracy measure of the model. We are reporting performance of all the models with validation / test dataset.

5.2.1 Evaluating Overall Performance of Models

Top 10 Best Models

Classifier	Embedding Name	AUC	Accuracy	Recall	Precision	F1
NB	fastTextWiki	0.80	0.76	0.78	0.75	0.76
TT	fastTextWiki	0.81	0.76	0.71	0.79	0.75
NB	IndicFT	0.77	0.74	0.70	0.76	0.73
LR	IndicFT	0.78	0.74	0.70	0.75	0.73
SVC	IndicFT	0.79	0.74	0.71	0.76	0.73
ADB	IndicFT	0.79	0.74	0.72	0.76	0.74
XGB	IndicFT	0.79	0.74	0.70	0.76	0.73
NB	Combined	0.79	0.74	0.76	0.74	0.75
PyrotchTT	mBERT	0.80	0.74	0.69	0.76	0.72
SVC	fastTextWiki	0.81	0.74	0.67	0.79	0.72

Table 5: Top 10 Best Models

Bottom 10 Worse Models

Classifier	Embedding Name	AUC	Accuracy	Recall	Precision	F1
ADB	mBERT	0.56	0.53	0.67	0.52	0.59
Perceptron	Word2Vec	0.52	0.52	0.47	0.52	0.49
NB	mBERT	0.55	0.52	0.30	0.53	0.38
DT	BOW	0.56	0.52	0.51	0.53	0.52
NB	BOW	0.58	0.52	0.39	0.52	0.45
Perceptron	mBERT	0.50	0.50	0.24	0.51	0.33
Perceptron	Lexical	0.50	0.50	0.00	0.00	0.00
Perceptron	Combined	0.50	0.50	0.01	0.50	0.02
NB	Word2Vec	0.64	0.50	0.95	0.50	0.66
NB	fastText	0.66	0.50	0.95	0.50	0.66

Table 6: Bottom 10 Worse Models

Across all the 109 models developed using Task Transfer, different kind of embedding and classifiers used the best Accuracy score is 76% with Naïve Bayesian Classifier when fastTextWiki embedding are used. AUC score is another good metrics to compares models. AUC score is free from thresholds, which is used to optimize the model performance. The best AUC score is 81% which we get when SVC (linear) is used with fastTextWiki transferred embedding. Lexical features could not demonstrate results in top 10. One interesting thing to note is when fastTextWiki embedding, the best embedding, is combined with Lexical features then overall accuracy of the model drops 2%. The fastText embedding, which is created using fastText library and without transfer learning, is one of the worse performer when used with NB. Perceptron classifier does not work good no matter what kind of embedding are used.

5.2.2 Evaluating Task Transferred Models

Task Transfer Learning

Embedding Name	AUC	Accuracy	Recall	Precision	F1
fastTextWiki	0.81	0.76	0.71	0.79	0.75
mBERT (Pytorch)	0.80	0.74	0.69	0.76	0.72
IndicFT	0.81	0.74	0.71	0.76	0.74
mBERT (Transformer)	0.60	0.58	0.65	0.57	0.61
IndicBERT (Transformer)	0.61	0.58	0.63	0.57	0.60

Table 7: Metrics for Task Transfer Models

In task transfer learning experiments, we used four base models and fined tuned classification task using our train data. The fastTextWiki task transfer model gives the best accuracy 76%. We had an interesting observation that IndicBERT & mBERT are giving bad results when we use them for direct task transfer. This is more interesting because BERT is one of the best performing models for classification task in English language and IndicBERT is tuned for Hindi language. However, with different set of parameters when we try mBERT on GPU machine with Pytorch implementation then performance improves significantly from 58% accuracy to 74% accuracy.

5.2.3 Evaluating Embedding & Classifier based on Average Metrics of Models

Best Classifier – Average
All Metrics of All Classifier

Embedding	Avg AUC	Avg Acc	Avg Recall	Avg Prec.	Avg F1
IndicFT	0.77	0.72	0.67	0.75	0.71
Keras_Tokenizer	0.74	0.68	0.68	0.69	0.68
fastTextWiki	0.76	0.69	0.64	0.73	0.67
Word2Vec	0.64	0.59	0.74	0.58	0.64
fastText	0.64	0.58	0.75	0.57	0.64
Combined	0.74	0.68	0.59	0.70	0.62
IndicBERT	0.64	0.61	0.59	0.62	0.59
TFIDF	0.63	0.59	0.58	0.60	0.59
BOW	0.63	0.59	0.55	0.60	0.57
Lexical	0.67	0.62	0.56	0.58	0.56
mBERT	0.60	0.57	0.58	0.57	0.56

Table 8: Best Embedding - Average (All Metrics & Classifiers)

Best Embedding – Average
All Metrics of All Embedding

Classifier	Avg AUC	Avg Acc	Avg Recall	Avg Prec.	Avg F1
RNN	0.74	0.68	0.70	0.67	0.68
CNN	0.73	0.66	0.68	0.66	0.67
RFC	0.72	0.65	0.71	0.65	0.67
SVC	0.70	0.65	0.68	0.65	0.66
GBC	0.69	0.65	0.65	0.66	0.65
XGB	0.70	0.64	0.63	0.65	0.64
LGBM	0.70	0.65	0.63	0.65	0.64
LR	0.69	0.64	0.64	0.64	0.64
ADB	0.67	0.63	0.63	0.63	0.63
DT	0.62	0.60	0.66	0.60	0.63
NB	0.67	0.60	0.64	0.61	0.60
Perceptron	0.56	0.56	0.36	0.55	0.41

Table 9: Best Classifier - Average (All Metrics & Embedding)

When we average out the performance of embedding for all the classifiers, we find those models which has IndicFT embedding are giving best Accuracy & F1 score. On the other hand if we average out performance of the classifiers for all the embedding we find RNN gives the best accuracy of 68%.

5.2.4 Evaluating Transferred Embedding

Embedding Transfer: fastText Wiki

Classifier	AUC	Accuracy	Recall	Precision	F1
NB	0.80	0.76	0.78	0.75	0.76
TT	0.81	0.76	0.71	0.79	0.75
SVC	0.81	0.74	0.67	0.79	0.72
LR	0.81	0.72	0.66	0.75	0.70
XGB	0.78	0.71	0.64	0.74	0.69
RFC	0.79	0.71	0.65	0.74	0.69
LGBM	0.78	0.70	0.63	0.72	0.67
ADB	0.79	0.70	0.65	0.73	0.69
GBC	0.78	0.69	0.62	0.72	0.67
CNN	0.74	0.65	0.74	0.63	0.68
Perceptron	0.63	0.63	0.36	0.78	0.49
DT	0.64	0.63	0.63	0.63	0.63

Table 10: Embedding Transfer - fastText Wiki

Embedding Transfer: IndicFT

Classifier	AUC	Accuracy	Recall	Precision	F1
NB	0.77	0.74	0.70	0.76	0.73
LR	0.78	0.74	0.70	0.75	0.73
SVC	0.79	0.74	0.71	0.76	0.73
ADB	0.79	0.74	0.72	0.76	0.74
XGB	0.79	0.74	0.70	0.76	0.73
TT	0.81	0.74	0.71	0.76	0.74
DT	0.71	0.72	0.61	0.77	0.68
Perceptron	0.72	0.72	0.56	0.81	0.66
LGBM	0.79	0.72	0.67	0.74	0.70
GBC	0.79	0.72	0.67	0.75	0.71
RFC	0.79	0.72	0.68	0.75	0.71
CNN	0.71	0.66	0.65	0.66	0.66

Table 11: Embedding Transfer- IndicFT

Embedding Transfer: mBERT

Classifier	AUC	Accuracy	Recall	Precision	F1
DT	0.63	0.62	0.65	0.61	0.63
SVC	0.63	0.60	0.66	0.59	0.63
GBC	0.64	0.60	0.65	0.60	0.62
RFC	0.64	0.60	0.65	0.59	0.62
LR	0.61	0.58	0.68	0.57	0.62
LGBM	0.63	0.58	0.64	0.57	0.60
XGB	0.61	0.57	0.62	0.56	0.59
ADB	0.56	0.53	0.67	0.52	0.59
NB	0.55	0.52	0.30	0.53	0.38
Perceptron	0.50	0.50	0.24	0.51	0.33

Table 12: Embedding Transfer - mBERT

Embedding Transfer: IndicBERT

Classifier	AUC	Accuracy	Recall	Precision	F1
RFC	0.71	0.70	0.70	0.70	0.70
XGB	0.71	0.66	0.65	0.67	0.66
LGBM	0.69	0.64	0.58	0.67	0.62
GBC	0.68	0.62	0.58	0.63	0.60
DT	0.62	0.60	0.62	0.60	0.61
ADB	0.64	0.60	0.59	0.60	0.59
NB	0.61	0.59	0.67	0.58	0.62
LR	0.59	0.57	0.61	0.57	0.59
Perceptron	0.56	0.56	0.22	0.67	0.33
SVC	0.60	0.56	0.65	0.55	0.59

Table 13: Embedding Transfer- IndicBERT

Combined Embedding : fastTestWiki + Lexical Features

Classifier	AUC	Accuracy	Recall	Precision	F1
NB	0.79	0.74	0.76	0.74	0.75
GBC	0.78	0.72	0.66	0.74	0.70
XGB	0.80	0.71	0.65	0.74	0.69
LGBM	0.78	0.70	0.63	0.72	0.67
ADB	0.78	0.70	0.64	0.74	0.68
LR	0.80	0.70	0.61	0.74	0.67
RFC	0.80	0.70	0.63	0.74	0.68
SVC	0.75	0.68	0.70	0.67	0.68
DT	0.63	0.63	0.63	0.63	0.63
Perceptron	0.50	0.50	0.01	0.50	0.02

Table 14: Embedding Transfer- Combined Embedding

As mentioned earlier, we have used 5 transfer embedding techniques. On fastTextWiki embedding NB gives the best accuracy 76%. On IndicFT embedding NB gives the best accuracy 74%. Nor classifier could give more than 62% accuracy on mBERT embedding. The best accuracy of IndicBERT embedding is with RFC classifier, 70% accuracy.

5.2.5 Evaluating Non-Transferred Embedding

Word2Vec Embedding

Classifier	AUC	Accuracy	Recall	Precision	F1
LGBM	0.68	0.64	0.72	0.62	0.66
ADB	0.64	0.62	0.71	0.61	0.65
GBC	0.64	0.62	0.70	0.61	0.65
SVC	0.67	0.62	0.75	0.60	0.66
XGB	0.69	0.62	0.63	0.62	0.63
LR	0.64	0.61	0.76	0.58	0.66
DT	0.60	0.58	0.79	0.56	0.65
RFC	0.69	0.57	0.89	0.54	0.67
Perceptron	0.52	0.52	0.47	0.52	0.49
NB	0.64	0.50	0.95	0.50	0.66

Table 15: Word2Vec Embedding

TFIDF Embedding

Classifier	AUC	Accuracy	Recall	Precision	F1
GBC	0.63	0.62	0.57	0.64	0.60
SVC	0.68	0.62	0.58	0.64	0.61
Perceptron	0.60	0.60	0.61	0.60	0.60
LR	0.64	0.60	0.52	0.61	0.56
LGBM	0.66	0.60	0.54	0.62	0.58
RFC	0.66	0.60	0.56	0.61	0.58
DT	0.61	0.58	0.73	0.57	0.64
NB	0.60	0.56	0.53	0.56	0.54
ADB	0.60	0.56	0.55	0.56	0.56
XGB	0.65	0.56	0.59	0.56	0.58

Table 16: TFIDF Embedding

BOW Embedding

Classifier	AUC	Accuracy	Recall	Precision	F1
RFC	0.68	0.64	0.68	0.62	0.65
ADB	0.61	0.62	0.61	0.62	0.62
LGBM	0.65	0.62	0.56	0.63	0.59
GBC	0.65	0.62	0.55	0.63	0.59
SVC	0.69	0.62	0.61	0.63	0.62
XGB	0.69	0.62	0.59	0.62	0.61
LR	0.63	0.60	0.50	0.62	0.56
Perceptron	0.55	0.55	0.51	0.55	0.53
DT	0.56	0.52	0.51	0.53	0.52
NB	0.58	0.52	0.39	0.52	0.45

Table 17: BOW Embedding

fastText Embedding

Classifier	AUC	Accuracy	Recall	Precision	F1
XGB	0.66	0.64	0.61	0.64	0.63
GBC	0.63	0.62	0.74	0.59	0.66
LGBM	0.66	0.62	0.65	0.61	0.63
SVC	0.67	0.61	0.75	0.59	0.66
LR	0.65	0.58	0.83	0.55	0.66
Perceptron	0.56	0.56	0.65	0.56	0.60
ADB	0.61	0.56	0.52	0.56	0.54
RFC	0.71	0.56	0.90	0.54	0.67
DT	0.54	0.54	0.87	0.52	0.65
NB	0.66	0.50	0.95	0.50	0.66

Table 18: fastText Embedding

Lexical Feature Engineering

Classifier	AUC	Accuracy	Recall	Precision	F1
LGBM	0.69	0.66	0.70	0.64	0.67
GBC	0.71	0.66	0.71	0.65	0.68
SVC	0.72	0.66	0.69	0.66	0.67
RFC	0.72	0.66	0.75	0.64	0.69
LR	0.74	0.66	0.57	0.70	0.63
ADB	0.68	0.62	0.63	0.62	0.63
DT	0.64	0.61	0.60	0.61	0.61
XGB	0.64	0.60	0.63	0.59	0.61
NB	0.69	0.58	0.32	0.67	0.43
Perceptron	0.50	0.50	0.00	0.00	0.00

Table 19: Lexical Feature Engineering

None of the non-transfer embedding techniques could deliver good results. The best non-transfer embedding technique is Lexical, which gives the best accuracy 66%. The fastText non-transfer embedding gives the best accuracy 64%.

5.2.6 Evaluating Models: Transfer Learning vs No-Transfer

Top 10 – Transfer Learning (Task & Embedding) Models

TL Type	Embedding Name	Classifier	AUC	Accuracy	Recall	Precision	F1
EMB	fastTextWiki	NB	0.80	0.76	0.78	0.75	0.76
Task	fastTextWiki	TT	0.81	0.76	0.71	0.79	0.75
EMB	IndicFT	NB	0.77	0.74	0.70	0.76	0.73
EMB	IndicFT	LR	0.78	0.74	0.70	0.75	0.73
EMB	IndicFT	SVC	0.79	0.74	0.71	0.76	0.73
EMB	IndicFT	ADB	0.79	0.74	0.72	0.76	0.74
EMB	IndicFT	XGB	0.79	0.74	0.70	0.76	0.73
EMB	Combined	NB	0.79	0.74	0.76	0.74	0.75
Task	mBERT (Pytorch)	PyrotchTT	0.80	0.74	0.69	0.76	0.72
EMB	fastTextWiki	SVC	0.81	0.74	0.67	0.79	0.72

Table 20: Top 10- Transfer Learning (Task & Embedding) Models

Top 10 – No Transfer Learning Models

Embedding Name	Classifier	AUC	Accuracy	Recall	Precision	F1
Keras_Tokenizer	CNN	0.74	0.68	0.65	0.70	0.67
Keras_Tokenizer	RNN	0.74	0.68	0.70	0.67	0.68
Lexical	LGBM	0.69	0.66	0.70	0.64	0.67
Lexical	GBC	0.71	0.66	0.71	0.65	0.68
Lexical	SVC	0.72	0.66	0.69	0.66	0.67
Lexical	RFC	0.72	0.66	0.75	0.64	0.69
Lexical	LR	0.74	0.66	0.57	0.70	0.63
fastText	XGB	0.66	0.64	0.61	0.64	0.63
Word2Vec	LGBM	0.68	0.64	0.72	0.62	0.66
BOW	RFC	0.68	0.64	0.68	0.62	0.65

Table 21: Top 10- Non-Transfer Learning Models

We can clearing see the transfer learning is giving the best result. If we do not use any transfer learning techniques then results are far behind. The best score when we do transfer embedding & transfer task, both, is 76% accuracy while without any kind of transfer the best result is 68% accuracy.

5.2.7 Evaluating Models with Embedding Used

Naïve Bayesian

Embedding Name	AUC	Accuracy	Recall	Precision	F1
fastTextWiki	0.80	0.76	0.78	0.75	0.76
IndicFT	0.77	0.74	0.70	0.76	0.73
Combined	0.79	0.74	0.76	0.74	0.75
IndicBERT	0.61	0.59	0.67	0.58	0.62
Lexical	0.69	0.58	0.32	0.67	0.43
TFIDF	0.60	0.56	0.53	0.56	0.54
mBERT	0.55	0.52	0.30	0.53	0.38
BOW	0.58	0.52	0.39	0.52	0.45
Word2Vec	0.64	0.50	0.95	0.50	0.66
fastText	0.66	0.50	0.95	0.50	0.66

Table 22: Naive Bayesian with All Embeddings

With Naïve Baysian classifier fastTextWiki gives the best accuracy.The results of NB with IndicFT is not far behind. On NB no other embedding is doing good work.

Support Vector Machine (SVM)

Embedding Name	AUC	Accuracy	Recall	Precision	F1
IndicFT	0.79	0.74	0.71	0.76	0.73
fastTextWiki	0.81	0.74	0.67	0.79	0.72
Combined	0.75	0.68	0.70	0.67	0.68
Lexical	0.72	0.66	0.69	0.66	0.67
Word2Vec	0.67	0.62	0.75	0.60	0.66
TFIDF	0.68	0.62	0.58	0.64	0.61
BOW	0.69	0.62	0.61	0.63	0.62
fastText	0.67	0.61	0.75	0.59	0.66
mBERT	0.63	0.60	0.66	0.59	0.63
IndicBERT	0.60	0.56	0.65	0.55	0.59

Table 23: Support Vector Machine with All Embeddings

SVM classifier is giving same accuracy with IndicFT and fastTextWiki embedding. The results of the other embedding are 6% less accuracy.

Logistic Regression

Embedding Name	AUC	Accuracy	Recall	Precision	F1
IndicFT	0.78	0.74	0.70	0.75	0.73
fastTextWiki	0.81	0.72	0.66	0.75	0.70
Combined	0.80	0.70	0.61	0.74	0.67
Lexical	0.74	0.66	0.57	0.70	0.63
Word2Vec	0.64	0.61	0.76	0.58	0.66
BOW	0.63	0.60	0.50	0.62	0.56
TFIDF	0.64	0.60	0.52	0.61	0.56
mBERT	0.61	0.58	0.68	0.57	0.62
fastText	0.65	0.58	0.83	0.55	0.66
IndicBERT	0.59	0.57	0.61	0.57	0.59

Table 24: Logistic Regression with All Embeddings

LR performs the best with IndicFT embedding but fastTextWiki with LR is not far behind.

XG Boost

Embedding Name	AUC	Accuracy	Recall	Precision	F1
IndicFT	0.79	0.74	0.70	0.76	0.73
fastTextWiki	0.78	0.71	0.64	0.74	0.69
Combined	0.80	0.71	0.65	0.74	0.69
IndicBERT	0.71	0.66	0.65	0.67	0.66
fastText	0.66	0.64	0.61	0.64	0.63
Word2Vec	0.69	0.62	0.63	0.62	0.63
BOW	0.69	0.62	0.59	0.62	0.61
Lexical	0.64	0.60	0.63	0.59	0.61
mBERT	0.61	0.57	0.62	0.56	0.59
TFIDF	0.65	0.56	0.59	0.56	0.58

Table 25: XG Boost with All Embeddings

AdaBoost

Embedding Name	AUC	Accuracy	Recall	Precision	F1
IndicFT	0.79	0.74	0.72	0.76	0.74
Combined	0.78	0.70	0.64	0.74	0.68
fastTextWiki	0.79	0.70	0.65	0.73	0.69
BOW	0.61	0.62	0.61	0.62	0.62
Word2Vec	0.64	0.62	0.71	0.61	0.65
Lexical	0.68	0.62	0.63	0.62	0.63
IndicBERT	0.64	0.60	0.59	0.60	0.59
TFIDF	0.60	0.56	0.55	0.56	0.56
fastText	0.61	0.56	0.52	0.56	0.54
mBERT	0.56	0.53	0.67	0.52	0.59

Table 26: AdaBoost with All Embeddings

XGBoost , AdaBoost classifer performs the best with IndicFT.

Gradient Boost Classifier (GBC)

Embedding Name	AUC	Accuracy	Recall	Precision	F1
Combined	0.78	0.72	0.66	0.74	0.70
IndicFT	0.79	0.72	0.67	0.75	0.71
fastTextWiki	0.78	0.69	0.62	0.72	0.67
Lexical	0.71	0.66	0.71	0.65	0.68
TFIDF	0.63	0.62	0.57	0.64	0.60
fastText	0.63	0.62	0.74	0.59	0.66
Word2Vec	0.64	0.62	0.70	0.61	0.65
BOW	0.65	0.62	0.55	0.63	0.59
IndicBERT	0.68	0.62	0.58	0.63	0.60
mBERT	0.64	0.60	0.65	0.60	0.62

Table 27: Gradient Boost Classifier with All Embeddings

GBC classifer is performing equally well on Combined embedding and IndicFT.

Light Gradient Boost Model (LGBM)

Embedding Name	AUC	Accuracy	Recall	Precision	F1
IndicFT	0.79	0.72	0.67	0.74	0.70
fastTextWiki	0.78	0.70	0.63	0.72	0.67
Combined	0.78	0.70	0.63	0.72	0.67
Lexical	0.69	0.66	0.70	0.64	0.67
Word2Vec	0.68	0.64	0.72	0.62	0.66
IndicBERT	0.69	0.64	0.58	0.67	0.62
BOW	0.65	0.62	0.56	0.63	0.59
fastText	0.66	0.62	0.65	0.61	0.63
TFIDF	0.66	0.60	0.54	0.62	0.58
mBERT	0.63	0.58	0.64	0.57	0.60

Table 28: Light Gradient Boost Model with All Embeddings

LGBM classifier is performing the best on IndicFT but fastTextWiki embedding is not far behind.

Random Forest Classifier (RFC)

Embedding Name	AUC	Accuracy	Recall	Precision	F1
IndicFT	0.79	0.72	0.68	0.75	0.71
fastTextWiki	0.79	0.71	0.65	0.74	0.69
IndicBERT	0.71	0.70	0.70	0.70	0.70
Combined	0.80	0.70	0.63	0.74	0.68
Lexical	0.72	0.66	0.75	0.64	0.69
BOW	0.68	0.64	0.68	0.62	0.65
mBERT	0.64	0.60	0.65	0.59	0.62
TFIDF	0.66	0.60	0.56	0.61	0.58
Word2Vec	0.69	0.57	0.89	0.54	0.67
fastText	0.71	0.56	0.90	0.54	0.67

Table 29: Radom Forest Classifier with All Embeddings

RFC classifer work the best with IndicFT & fastTextWiki embeddings. But the results of IndicBERT embedding with RFC is not far behind.

Perceptron

Embedding Name	AUC	Accuracy	Recall	Precision	F1
IndicFT	0.72	0.72	0.56	0.81	0.66
fastTextWiki	0.63	0.63	0.36	0.78	0.49
TFIDF	0.60	0.60	0.61	0.60	0.60
IndicBERT	0.56	0.56	0.22	0.67	0.33
fastText	0.56	0.56	0.65	0.56	0.60
BOW	0.55	0.55	0.51	0.55	0.53
Word2Vec	0.52	0.52	0.47	0.52	0.49
mBERT	0.50	0.50	0.24	0.51	0.33
Lexical	0.50	0.50	0.00	0.00	0.00
Combined	0.50	0.50	0.01	0.50	0.02

Table 30: Perceptron with All Embeddings

Perceptron works the best with IndicFT, other embedding has 9% less accuracy than IndicFT with perceptron.

With perceptron IndicFT give the best F1 score but with AdaBoost non of the embedding is able to give more than 70% of F1 score.

Decision Tree

Embedding Name	AUC	Accuracy	Recall	Precision	F1
IndicFT	0.71	0.72	0.61	0.77	0.68
Combined	0.63	0.63	0.63	0.63	0.63
fastTextWiki	0.64	0.63	0.63	0.63	0.63
mBERT	0.63	0.62	0.65	0.61	0.63
Lexical	0.64	0.61	0.60	0.61	0.61
IndicBERT	0.62	0.60	0.62	0.60	0.61
Word2Vec	0.60	0.58	0.79	0.56	0.65
TFIDF	0.61	0.58	0.73	0.57	0.64
fastText	0.54	0.54	0.87	0.52	0.65
BOW	0.56	0.52	0.51	0.53	0.52

Table 31: Decision Tree with All Embeddings

Decision tree on IndicFT give best accuracy 72%. But on fastTextWiki its performance is 9% less.

5.2.8 Evaluating CNN & RNN Models

CNN and RNN

Classifier	Embedding Name	AUC	Accuracy	Recall	Precision	F1
CNN	Keras_Tokenizer	0.74	0.68	0.65	0.70	0.67
RNN	Keras_Tokenizer	0.74	0.68	0.70	0.67	0.68
CNN	IndicFT	0.71	0.66	0.65	0.66	0.66
CNN	fastTextWiki	0.74	0.65	0.74	0.63	0.68

Table 32: CNN & RNN Model Results

Results of CNN & RNN classification models is not comparable to other models discussed earlier. Even if we use the best embedding and transfer it to our CNN model, we are not getting more than 66% accuracy.

5.3 Evaluation & Sampling Methods of Results

In all our experiments we calculated five metrics namely Accuracy, Recall, Precision, F1 Score and AUC score. But all the tables are shorted based on the accuracy score, highest to lowest. Although we displayed results but on accuracy score we considered top 2 performing model as the best models.

5.4 Comparing Results with Other Works

However, did not find any work which has been on Hinglish language and using twitter and normal blog text together for sarcasm work yet we are putting a table below to demonstrate other work and compare the progress made by our work

#	Paper	Language, Text Type	Metrics
Sarcasm Detection Work in English Language			
1	Irony Detection in Twitter: The Role of Affective Content. (Fa ́ıas et al., 2016)	English, Twitter	Acc: 73-96% depends upon datasets and classifier.
2	Natural Language Processing Based Features for Sarcasm Detection: An Investigation Using Bilingual Social Media Texts. (Suhaimin et al., 2017)	English, Twitter	Acc: 82.5%
3	Semantics-aware BERT for Language Understanding. (Zhang et al., 2020b)	English, Normal Text	Acc: 94.6% on Large dataset of SST2
4	Multi-Rule Based Ensemble Feature Selection Model for Sarcasm Type Detection in Twitter. (Sundararajan and Palanisamy, 2020)	English, Twitter	Acc: 86.61% to 99.79% Depending upon the type of sarcasm. Final classifier is RF
5	Sarcasm Detection in Typo-graphic Memes (Kumar and Garg, 2019)	English, Instagram Images	Acc: 73.25% to 87.95% depending upon the classifier used.
6	Sarcasm detection on twitter : A Behavioural Modeling Approach.	English, Tweet	Acc: 83.46%

	(Rajadesingan et al., 2015)		
7	Lexicon-Based Sentiment Analysis in the Social Web. (Asghar et al., 2014)	English, Tweet	Acc: 95.24%
8	Harnessing Context Incongruity for Sarcasm Detection. (Joshi et al., 2015)	English, Tweet	F1: 61%
9	Contextualized Sarcasm Detection on Twitter (Bamman and Smith, 2015)	English, Tweet	Acc: 85.1%
10	Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. (Turney, 2002)	English, Opinion Survey of Products	Acc: 74.39%
11	Towards Multimodal Sarcasm Detection: An Obviously Perfect Paper (Castro et al., 2020)	English, Clips of YouTube, TV Shows, Transcription	F1: 71.8%
12	A Transformer-based approach to Irony and Sarcasm detection (Potamias et al., 2020)	English, Irony/SemVal-2018-Task, Reddit SARC2.0 politics, Riloff Sarcastic Dataset	Acc: 85% to 94% depending upon dataset
13	Detecting Sarcasm is Extremely Easy ;-) (Parde and Nielsen, 2018)	English, Tweet, Amazon product reviews	F1: 59% (Twitter) F1: 78% (Amazon)
14	CARER: Contextualized Affect Representations for Emotion Recognition (Saravia et al., 2020)	English, Tweets	Acc: 81% with CARER
15	The perfect solution for detecting sarcasm in tweets #not (Liebrecht et al., 2013)	Dutch, Tweets	AUC: 77%
16	A2Text-net: A novel deep neural network for sarcasm detection (Liu et al., 2019a)	English, Tweet, News Headlines, Reddit	F1: 71% - 90% depending upon dataset with A2Text classifer
17	Sarcasm as contrast between a positive sentiment and negative situation (Riloff et al., 2013)	English, Tweet	F1: 51%

18	Exploring the fine-grained analysis and automatic detection of irony on Twitter (Van Hee et al., 2018)	English, Tweet	Acc: 67.54% (SVM) Acc: 68.27% (LSTM)
19	Exploiting Emojis for Sarcasm Detection (Subramanian et al., 2019)	English, Twitter, Facebook	F1: 89.36% (Twitter) F1: 97.97% (facebook)
20	A novel automatic satire and irony detection using ensembled feature selection and data mining. (Ravi and Ravi, 2017)	English, Newswire, Satire news articles, Amazon	F1: 96.58% (L+T+D features) + GR feature selector + SVM RBF Classifier
21	Automatic Satire Detection: Are You Having a Laugh? (Burfoot and Baldwin, 2009)	English, Newswire and Satire news articles	F1: 79.8%
22	Semi-supervised recognition of sarcastic sentences in twitter and Amazon (Davidov et al., 2010)	English, Twitter, Amazon	F1: 78% Amazon F1: 83% Twitter
23	Identifying Sarcasm in Twitter: A Closer Look. In (González-Ibáñez et al., 2011)	English, Twitter	Acc: 55.59% to 75.78% depending upon tweet format.
Sarcasm Detection Work in Hindi Language			
1	Sentiment Analysis of Hindi Review based on Negation and Discourse Relation. (Mittal and Agarwal, 2013)	Hindi, Movie Reviews	Acc: 80.21%
2	A Sentiment Analyzer for Hindi Using Hindi Senti Lexicon. (Sharma et al., 2014)	Hindi, Movie Reviews, Product Reviews	Acc: 85 to 89.5%
3	Sarcasm Detection in Hindi sentences using Support Vector (Desai and Dave, 2016)	Hindi, various online sources (using polarity levelled corpora)	Acc: 84%
4	Sentiment Analysis in a Resource Scarce Language: Hindi. (Jha et al., 2016)	Hindi, Movie Reviews	Acc: 92.2% to 100% depending upon unigram or bigram feature and classifer
5	Harnessing Online News for Sarcasm Detection in Hindi Tweets (Bharti et al., 2017)	Hindi, Tweets	Acc: 79.4%

6	Context-based Sarcasm Detection in Hindi Tweets. (Bharti et al., 2018)	Hindi, Tweets	Acc: 87%
7	A Corpus of English-Hindi Code-Mixed Tweets for Sarcasm Detection (Swami et al., 2018)	Hindi-English, Tweets	Acc: 78.4% with RF
8	BHAAV- A Text Corpus for Emotion Analysis from Hindi Stories (Kumar et al., 2019)	Hindi, Short stories	Acc: 62%
9	**Sarcasm Detection in Hinglish Language**	**Hinglish Language, Twitter + Blog Text**	**Accuracy: 76%, Recall: 78%, Precision: 75%, F1: 76%, AUC: 80%**

Table 33: Model Performance Comparison

5.5 Summary

We compared the 5 metrics of 109 models. These models were developed using different embeddings and classifiers. Because our dataset is balanced so we took accuracy as a metrics to measure the performance of these models. However we have also considered other metrics like F1, ROC so that is easier to the performance with the models developed by other researchers. Those researchers sometimes have used other metrics to report the performance of their models. Transferred Learning is the most powerful techninuqe to produce good results on Hinglish language sarcasm detection. Both, the task transfer and embedding transfer gives good results. Naïve Bayesian classifier is the best suited for sarcasm detection provided a good embedding is used to build the model. CNN, RNN could not produce good results even with the best embedding transfer to the network. We think we need more complex architecture to get good results or more tuning of hypter parameters is required.

CHAPTER 6 : CONCLUSIONS AND RECOMMENDATIONS

6.1 Introduction

We started our work with researching around the meaning of sarcasm in the Indian context. In the context of NLP and ML work we established what is the meaning of Hinglish language. How sarcasm is expressed on various social media platform using Hinglish language is also explored in this work. We explored the existing systems developed to detect sarcasm using machine learning and found that there are some attempts to develop systems but those are limited to pure Hindi and limited to twitter text. We did not find any work which has been done to detect sarcasm in Hinglish language and which is beyond twitter text. We developed a balanced sarcasm dataset of 2000 Hinglish sentences using blog text and tweets. This dataset was developed with the help of three annotators. This dataset is available for other related research work or those who want to take this work to next stage of improvement. We experimented various embedding techniques and classification techniques along with transfer learning.

6.2 Discussion & Conclusion

Regarding Transfer Embedding, we want to report that when Transfer Embedding is combined with classical machine learning algorithm then it gives the best result with Naïve Bayesian classifier. Two embedding transfer fastTextWiki and IndicFT both gives competitive results 76% and 74% accuracy respectably with NB classifier. We need to keep in mind both are fastText based embedding. As discussed earlier, the fastText is subword based embedding technique. Thus, we report that fastText based pretrained embedding is the best suitable for Hinglish data.

Regarding non-transfer Embedding, we want to report that all other models which were created using our own embedding could not perform good with any classifier used. Even our embedding which we create using lexical features could not give good results. The best result of Lexical features is with LGBM classifier. Here we get the 66% accuracy and with other classifiers we get as low as 50% accuracy, which is as good as wild guess.

Regarding Task Transfer, we want to report that we get the highest accuracy 76% when fastTextWiki pretrained model is used for classification. When mBERT is implemented with Pytorch for task transfer and IndicFT is implemented with transformer for task transfer we get

74% accuracy. But results are not good when IndicBERT or mBERT is implemented with Transformer, we get 58% accuracy.

Regarding Classifier, we observed that for Hinglish Language sarcasm detection NB & SVC are the best classifier when they are combined with embedding transfer.

We experimented with various options of transliteration so that we get the complete sentence in the Devanagari script, but we could not get good quality transliteration. We tried libraries like indicnlp.acronym_transliterator, indic_transliteration.sanscript, indicnlp.transliterate.unicode_transliterate.ItransTransliterator, indicnlp.syllable.syllabifier, indicnlp.script.indic_scripts, aksharamukha.transliterate, but nothing helped us in consistent transliteration of the text from Roman from Devanagari. Transliteration challenges in Hinglish languages are discussed in detail in a separate article. Those who are interested can read here. This challenge lead to developing all embeddings without the transliteration of the input text.

As of today, we did not find pretrained embedding which is created using huge Hinglish language corpora. We have limited time, hardware, linguistic experts and other resources which leads to small corpora for training. Due to this reason metrics are not giving results which can lead to a product to be used for commercial deployment. However, if we compare the results of our system with the systems developed for other languages our results are encouraging.

We further analysed prediction results of two different type of text and found prediction results on test data for blog text is better than on twitter text. This may be because twitter text is not as regular text as blog text. Twitter text has lots of emoticon, mix of script, mix of language, spelling mistate while that is not the case with regular blog text.

6.3 Contribution to knowledge

Broadly our work has done four contributions.

1. We created sarcasm dataset in Hinglish languages of 2000 sentences, and it can be used by other researchers to create their models or expend this dataset.
2. We identified good embedding (fastTextWiki and IndicFT) for finetuning, these are more effective for Hinglish language sarcasm detection work.
3. We established even if combine embedding with classical ML classifier we get good results, provided we have good embedding. So, the task transfer is not mandatory.

4. We identified, if we do task transfer then which models can be used for getting good results on Hinglish Language sarcasm detection work.

6.4 Future Recommendations

6.4.1 Dataset

Our dataset has only 2000 sentences. To make a stable model we need more data for this sarcasm classification task. Hence, in future work we should focus on expending the dataset. We should include more Hinglish language text. Create a balance dataset from twitter and blog text.

6.4.2 Classifiers & Task Transfer

We have tried task transfer and we also used classical ML classifier with embedding transfer, embedding creation and lexical feature. We got good results even with task transfer. We would recommend using more sophisticated models like GPT3 in future experiments and conduct those experiments on GPU/ TPU machines with more data. As NB & SVM is giving good results so we would recommend to use these classifiers with embedding transfer.

6.4.3 Embedding

We used mBERT, fastTextWiki, IndicFT and IndicBERT to finetune and transfer embedding. These models are primarily trained on the corpus of Devanagari script and Hindi language. In the real world around text us is not purely in Devanagari script and Hindi language. Therefore, we cannot rely on the embedding of these models for good results. Hence, we need to collect a huge size corpus from social media communication of Hinglish language and create an embedding model.

6.4.4 Language Treatment of words in the Sentence

We know Roman typing is much easy compare to typing in Devanagari therefore many time people use Roman letters in between the sentence. This is true especially if it is name of politician, film actor, place name, movie name, event name (#AmitShah, #Modi, #Khan, #India #Bollywood, #Delhi, #Karnataka #Yogi, #Dangal #Deepoutsav) etc. If we are using transfer learning for a model created using non-Hinglish corpus then we need to transliterate all these words into Devanagari script. We attempted this work in our project, but we could not do this

successfully and in future we need to find or develop a library which can do this transliteration in a most reliable way. We think this transliteration can be summarised as below.

1. **Hindi word in Devanagari**: No change required.
2. **Non-Hindi Indian words in Devanagari**: Words from other language like Urdu, Punjabi, Marathi like खत्म, खल्लास, गजल, सोन्देश, मसवरा दास्तां खबर should be left as is.
3. **English words in Devanagari**: English words written in Devanagari like "राइस" "विन" "ग्रेट" should be left as is.
4. **Hindi/Non-English word in Roman script words**: Hindi/Non-English words written in roman scripts like "Aap to Mahan hai", "tussi great ho ji" should be transliterated to Devanagari. So, it will be like "आप तो महान हैं"
5. **English words in Roman**: English words in Roman like "friend", "love" in between Hinglish sentence e.g. "मैं अपने friend को love करता हूं", we should transliterate "friend" to Devanagari and we should not try to translate this. So, new sentence should be ". "मैं अपने फ्रेंड को लव करता हूं" and not like this "मैं अपने मित्र को प्यार करता हूं"

REFERENCES

[1.] Anggraini, S.D., (2014) *A Pragmatic Analysis Of Humor In Modern Family*.

[2.] Asghar, M.Z., Kundi, F.M., Khan, A. and Ahmad, S., (2014) Lexicon-Based Sentiment Analysis in the Social Web. *J. Basic. Appl. Sci. Res*, 46, pp.238–248.

[3.] Bamman, D. and Smith, N.A., (2015) Contextualized sarcasm detection on twitter. *Proceedings of the 9th International Conference on Web and Social Media, ICWSM 2015*, pp.574–577.

[4.] Bharti, S.K., Babu, K.S. and Raman, R., (2018) Context-based Sarcasm Detection in Hindi Tweets. *2017 9th International Conference on Advances in Pattern Recognition, ICAPR 2017*, pp.410–415.

[5.] Bharti, S.K., Sathya Babu, K. and Jena, S.K., (2017) Harnessing Online News for Sarcasm Detection in Hindi Tweets. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10597 LNCS, pp.679–686.

[6.] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., (2017) Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, [online] 5, pp.135–146. Available at: http://www.isthe.com/chongo/tech/comp/fnv.

[7.] Van Brunt, J., (1987) A Closer Look at Fermentors and Bioreactors. *Nature Biotechnology*, [online] 511, pp.1133–1138. Available at: http://www.nature.com/doifinder/10.1038/nbt1187-1133 [Accessed 29 Aug. 2020].

[8.] Burfoot, C. and Baldwin, T., (2009) Automatic satire detection: Are you having a laugh? In: *ACL-IJCNLP 2009 - Joint Conf. of the 47th Annual Meeting of the Association for Computational Linguistics and 4th Int. Joint Conf. on Natural Language Processing of the AFNLP, Proceedings of the Conf.* [online] pp.161–164. Available at: http://search.cpan.org/perldoc? [Accessed 22 Aug. 2020].

[9.] Castro, S., Hazarika, D., Pérez-Rosas, V., Zimmermann, R., Mihalcea, R. and Poria, S., (2020) Towards multimodal sarcasm detection (an obviously perfect paper). *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pp.4619–4629.

[10.] Clark, K., Luong, M.-T., Le, Q. V. and Manning, C.D., (2020) ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. [online] Available at:

https://github.com/google-research/ [Accessed 28 Aug. 2020].

[11.] Davidov, D., Tsur, O. and Rappoport, A., (2010) Semi-supervised recognition of sarcastic sentences in twitter and Amazon. *CoNLL 2010 - Fourteenth Conference on Computational Natural Language Learning, Proceedings of the Conference*, July, pp.107–116.

[12.] Desai, N.P. and Dave, A.D., (2016) Sarcasm Detection in Hindi sentences using Support Vector machine. *International Journal of Advance Research in Computer Science and Management Studies*, [online] 47, pp.8–15. Available at: www.ijarcsms.com.

[13.] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1Mlm, pp.4171–4186.

[14.] Fańias, D.I.H., Patti, V. and Rosso, P., (2016) Irony detection in twitter: The role of affective content. *ACM Transactions on Internet Technology*, [online] 163, pp.1–24. Available at: http://dx.doi.org/10.1145/2930663 [Accessed 17 Aug. 2020].

[15.] Fawcett, T., (2004) ROC graphs: Notes and practical considerations for researchers. *Machine learning*, 311, pp.1–38.

[16.] Gaikwad, V. and Haribhakta, Y., (2020) Adaptive glove and fasttext model for Hindi word embeddings. *ACM International Conference Proceeding Series*, pp.175–179.

[17.] González-Ibáñez, R., Muresan, S. and Wacholder, N., (2011) Identifying Sarcasm in Twitter: A Closer Look. In: *Proceedings ofthe 49th Annual Meeting ofthe Association for Computational Linguistics:shortpapers*. [online] pp.581–586. Available at: http://www.vidarholen.net/contents/interjections/ [Accessed 16 Aug. 2020].

[18.] Van Hee, C., Lefever, E. and Hoste, V., (2018) Exploring the fine-grained analysis and automatic detection of irony on Twitter. *Language Resources and Evaluation*, 523, pp.707–731.

[19.] Jha, V., N, M., Shenoy, P.D. and K R, V., (2016) Sentiment Analysis in a Resource Scarce Language:Hindi. *International Journal of Scientific & Engineering Research*, 79, pp.968–980.

[20.] Joshi, A., Bhattacharyya, P. and Carman, M.J., (2018) Investigations in computational sarcasm. *Cognitive Systems Monographs*, 37, pp.137–143.

[21.] Joshi, A., Sharma, V. and Bhattacharyya, P., (2015) Harnessing context incongruity for sarcasm detection. *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for*

Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference, 22003, pp.757–762.

[22.] Kakwani, D., Kunchukuttan, A., Golla, S., N.C., G., Bhattacharyya, A., Khapra, M.M. and Kumar, P., (2020) IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In: *Findings of EMNLP*.

[23.] Kumar, A. and Garg, G., (2019) Sarc-M : Sarcasm Detection in Typo-graphic Memes. In: *International Conference on Advanced Engineering, Science, Management and Technology – 2019 (ICAESMT19) Sarc-M:* pp.1–8.

[24.] Kumar, Y., Mahata, D., Aggarwal, S., Chugh, A., Maheshwari, R. and Shah, R.R., (2019) *BHAAV- A Text Corpus for Emotion Analysis from Hindi Stories*. Available at: http://arxiv.org/abs/1910.04073%0Ahttp://dx.doi.org/10.5281/zenodo.3457467.

[25.] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R., (2019) ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. [online] Available at: https://github.com/google-research/ALBERT. [Accessed 28 Aug. 2020].

[26.] Lee, C.J., Katz, A.N., Lee, C.J. and Katz, A.N., (2009) The Differential Role of Ridicule in Sarcasm and Irony The Differential Role of Ridicule in Sarcasm and Irony. 6488May 2015, pp.37–41.

[27.] Liebrecht, C., Kunneman, F. and Bosch, A. Van den, (2013) The perfect solution for detecting sarcasm in tweets #not. [online] June, pp.29–37. Available at: http://www.aclweb.org/anthology/W13-1605.

[28.] Liu, L., Priestley, J.L., Zhou, Y., Ray, H.E. and Han, M., (2019a) A2Text-net: A novel deep neural network for sarcasm detection. *Proceedings - 2019 IEEE 1st International Conference on Cognitive Machine Intelligence, CogMI 2019*, December, pp.118–126.

[29.] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V., (2019b) RoBERTa: A Robustly Optimized BERT Pretraining Approach. [online] Available at: https://github.com/pytorch/fairseq [Accessed 28 Aug. 2020].

[30.] Mittal, N. and Agarwal, B., (2013) Sentiment Analysis of Hindi Review based on Negation and Discourse Relation. *Sixth International Joint Conference on Natural Language Processing*, [online] October, pp.57–62. Available at: http://www.aclweb.org/website/old_anthology/W/W13/W13-43.pdf#page=57.

[31.] Nozza, D., Fersini, E. and Messina, E., (2016) Unsupervised Irony Detection: A Probabilistic Model with Word Embeddings. In: *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. [online] SCITEPRESS - Science and Technology Publications, pp.68–76. Available at: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006052000680076 [Accessed 16 Aug. 2020].

[32.] Parde, N. and Nielsen, R., (2018) Detecting Sarcasm is Extremely Easy ;-). pp.21–26.

[33.] Pires, T., Schlinger, E. and Garrette, D., (2020) How multilingual is multilingual BERT? In: *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*. [online] pp.4996–5001. Available at: https://github.com/google-research/bert [Accessed 11 Oct. 2020].

[34.] Potamias, R.A., Siolas, G. and Stafylopatis, A.G., (2020) A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*.

[35.] Qi, P., Dozat, T., Zhang, Y. and Manning, C.D., (2018) Universal Dependency Parsing from Scratch. In: *Proceedings of the {CoNLL} 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. [online] Brussels, Belgium: Association for Computational Linguistics, pp.160–170. Available at: https://nlp.stanford.edu/pubs/qi2018universal.pdf.

[36.] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J., (2019) Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, [online] 21, pp.1–67. Available at: http://jmlr.org/papers/v21/20-074.html. [Accessed 28 Aug. 2020].

[37.] Rajadesingan, A., Zafarani, R. and Liu, H., (2015) Sarcasm detection on twitter:A behavioral modeling approach. *WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, pp.97–106.

[38.] Ramos, J., (2003) Using TF-IDF to Determine Word Relevance in Document Queries. In: *Proceedings of the first instructional conference on machine learning*. [online] pp.133–142. Available at: https://sites.google.com/site/caonmsu/ir/UsingTFIDFtoDetermineWordRelevanceinDocumentQueries.pdf [Accessed 29 Aug. 2020].

[39.] Ravi, K. and Ravi, V., (2017) A novel automatic satire and irony detection using ensembled feature selection and data mining. *Knowledge-Based Systems*, [online] 120, pp.15–33. Available at: http://dx.doi.org/10.1016/j.knosys.2016.12.018.

[40.] Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N. and Huang, R., (2013) Sarcasm as contrast between a positive sentiment and negative situation. *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, October, pp.704–714.

[41.] Romano, S., (2020) *Multilingual Transformers - Towards Data Science*. [online] Available at: https://towardsdatascience.com/multilingual-transformers-ae917b36034d [Accessed 28 Aug. 2020].

[42.] Sanh, V., Debut, L., Chaumond, J. and Wolf, T., (2019) DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. [online] Available at: https://github.com/huggingface/transformers [Accessed 28 Aug. 2020].

[43.] Saravia, E., Toby Liu, H.C., Huang, Y.H., Wu, J. and Chen, Y.S., (2020) Carer: Contextualized affect representations for emotion recognition. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pp.3687–3697.

[44.] Sharma, D.S., Sangal, R., Pawar, J.D., Sharma, R. and Bhattacharyya, P., (2014) A Sentiment Analyzer for Hindi Using Hindi Senti Lexicon. In: *NLP Association of India*. NLPAI, pp.150–155.

[45.] Sinha, R.M.K. and Thakur, A., (2005) Machine Translation of Bi-lingual Hindi-English (Hinglish) Text. *10th Machine Translation summit (MT Summit X)*, pp.149–156.

[46.] Subramanian, J., Sridharan, V., Shu, K. and Liu, H., (2019) Exploiting emojis for sarcasm detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11549 LNCSApril, pp.70–80.

[47.] Suhaimin, M.S.M., Hijazi, M.H.A., Alfred, R. and Coenen, F., (2017) Natural language processing based features for sarcasm detection: An investigation using bilingual social media texts. *ICIT 2017 - 8th International Conference on Information Technology, Proceedings*, pp.703–709.

[48.] Sundararajan, K. and Palanisamy, A., (2020) Multi-rule based ensemble feature selection model for sarcasm type detection in Twitter. *Computational Intelligence and Neuroscience*, 2020.

[49.] Swami, S., Khandelwal, A., Singh, V., Akhtar, S.S. and Shrivastava, M., (2018) A Corpus of English-Hindi Code-Mixed Tweets for Sarcasm Detection. [online] pp.1–9. Available at: http://arxiv.org/abs/1805.11869.

[50.] Turney, P.D., (2002) Thumbs Up or Thumbs Down? Semantic Orientation Applied to

Unsupervised Classification of Reviews. [online] July, pp.417–424. Available at: http://arxiv.org/abs/cs/0212032.

[51.] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., (2017) Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-DecemNips, pp.5999–6009.

[52.] Wang, Q., Xu, J., Chen, H. and He, B., (2017) Two improved continuous bag-of-word models. In: *Proceedings of the International Joint Conference on Neural Networks*. [online] Institute of Electrical and Electronics Engineers Inc., pp.2851–2856. Available at: http://ieeexplore.ieee.org/document/7966208/ [Accessed 29 Aug. 2020].

[53.] Wikipage, (2020a) *Demographics of India - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/Demographics_of_India [Accessed 29 Aug. 2020].

[54.] Wikipage, (2020b) *Internet in India - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/Internet_in_India [Accessed 29 Aug. 2020].

[55.] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. and Le, Q. V, (2019) XLNet: Generalized Autoregressive Pretraining for Language Understanding. In: *33rd Conference on Neural Information Processing Systems (NeurIPS*. [online] Available at: https://github.com/zihangdai/xlnet [Accessed 28 Aug. 2020].

[56.] Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J. and Dolan, B., (2020a) DIALOGPT : Large-Scale Generative Pre-training for Conversational Response Generation. [online] pp.270–278. Available at: https://github.com/mgalley/ [Accessed 28 Aug. 2020].

[57.] Zhang, Z., Wu, Y., Zhao, H., Li, Z., Zhang, S., Zhou, X. and Zhou, X., (2020b) Semantics-Aware BERT for Language Understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 3405, pp.9628–9635.

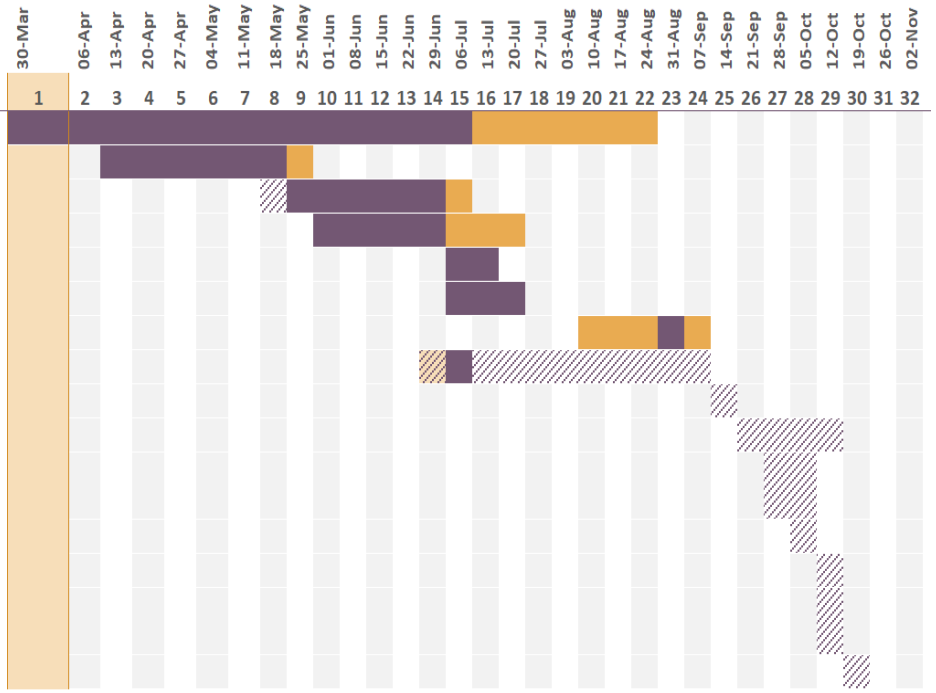
APPENDIX A: LIST OF OTHER DOCUMENTS

These are the documents prepared during this project but for the purpose of brevity there are not part of main document. Those who are interested can refer them in github.

1. History_AutoSarcasmDetection.pdf
2. Summary-of-Sarcasm-Papers.pdf
3. Dataset-cleaning-steps.pdf
4. Datasource-links.pdf
5. All Metrics – All Models
6. Transliteration Challenges in Hinglish Language

APPENDIX B: RESEARCH PLAN

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Litration Review	1	15	1	22	100%
Research Topic Selection	3	6	3	7	100%
Research Proposal Writing	8	7	9	7	100%
Create a dataset with of 2000 Sent.	10	5	10	8	100%
Research Embedding for Hinglish	15	2	15	2	100%
Finalize Approach	15	3	15	3	100%
Prepare and Submit Interim Report	23	1	20	5	100%
Develop 4+ Models	15	10	15		10%
Evaluate Performance of Models	25	1			
Write Final Desertation	26	4			
Review Desertation with Guide (1st Review)	27	2			
Compare Models/ Conclusion	28	1			
Prepare a final Presentation	29	1			
Review Desertation with Guide (2nd Review)	29	1			
Complete report	30	1			



APPENDIX C: SARCASM DETECTION SYSTEMS RESULTS OF PAST WORK.

Sno	Year	Authors	Model	Features	Metrics
1	2002	(Turney, 2002)	Rule based	LFS	Acc: 74.39%
2	2009	(Burfoot and Baldwin, 2009)	Classical ML	LFS	F1: 79.8%
3	2010	(Pak and Paroubek, 2010)	Classical ML	LFS	Not Mentioned
4	2010	(Davidov et al., 2010)	Classical ML	LFS	F1: 78% Amazon
5	2010	(Davidov et al., 2010)	Classical ML	LFS	F1: 83% Twitter
6	2011	(González-Ibáñez et al., 2011)	Classical ML	LFS	Acc: 55.59% to 75.78% depending upon tweet format.
7	2013	(Mittal and Agarwal, 2013)	Rule Based	LFS	Acc: 80.21%
8	2013	(Liebrecht et al., 2013)	Ruled Based	LFS	AUC: 77%
9	2013	(Riloff et al., 2013)	Classical ML	LFS	F1: 51%
10	2014	(Asghar et al., 2014)	Rule based	LFS	Acc: 95.24%
11	2014	(Sharma et al., 2014)	Rule Based	LFS	Acc: 85 to 89.5%
12	2015	(Rajadesingan et al., 2015)	Classical ML	LFS	Acc: 83.46%
13	2015	(Joshi et al., 2015)	Classical ML	LFS	F1: 61%
14	2015	(Bamman and Smith, 2015)	Classical ML	LFS	Acc: 85.1%
15	2016	(Fárias et al., 2016)	Classical ML	LFS	Acc: 73-96% depends upon datasets and classifier.
16	2016	(Jha et al., 2016)	Classical ML	LFS	Acc: 92.2% to 100% depending upon unigram or bigram feature and classifer
17	2016	(Desai and Dave, 2016)	Classical ML	LFS	Acc: 84%
18	2017	(Suhaimin et al., 2017)	Classical ML	LFS	Acc: 82.5%

19	2017	(Bharti et al., 2017)	Rule Based	LFS	Acc: 79.4%
20	2017	(Ravi and Ravi, 2017)	Classical ML	LFS	F1: 96.58% (L+T+D features) + GR feature selector + SVM RBF Classifier
21	2018	(Bharti et al., 2018)	Rule Based	LFS	Acc: 87%
22	2018	(Parde and Nielsen, 2018)	Classical ML	LFS	F1: 59% (Twitter)
23	2018	(Parde and Nielsen, 2018)	Classical ML	LFS	F1: 78% (Amazon)
24	2018	(Swami et al., 2018)	Classical ML	LFS	Acc: 78.4% with RF
25	2018	(Van Hee et al., 2018)	Classical ML	LFS	Acc: 67.54% (SVM)
26	2018	(Van Hee et al., 2018)	Classical ML	LFS	Acc: 68.27% (LSTM)
27	2019	(Kumar and Garg, 2019)	Classical ML	LFS	Acc: 73.25% to 87.95% depending upon the classifier used.
28	2019	(Kumar et al., 2019)	Classical ML	Both	Acc: 62%
29	2019	(Subramanian et al., 2019)	GRU	LFS	F1: 89.36% (Twitter)
30	2019	(Subramanian et al., 2019)	GRU	LFS	F1: 97.97% (facebook)
31	2019	(Liu et al., 2019a)	Classical + CNN	LFS	F1: 71% - 90% depending upon dataset with A2Text classifer
32	2020	(Zhang et al., 2020b)	CNN	LFS	Acc: 94.6% on Large dataset of SST2
33	2020	(Sundararajan and Palanisamy, 2020)	Classical ML	LFS	Acc: 86.61% to 99.79% Depending upon the type of sarcasm. Final classifier is RF
34	2020	(Castro et al., 2020)	Classical ML	LFS	F1: 71.8%
35	2020	(Potamias et al., 2020)	Transformer	Embedding	Acc: 85% to 94% depending upon dataset
36	2020	(Saravia et al., 2020)	CNN	Both	Acc: 81% with CARER

APPENDIX D: ALL METRICES OF ALL MODELS

The table below summarises the performance of all 109 models developed during our experiments. Pink color shows top two models with that metrics.

Model #	TL	TL Type	Embedding Name	Classifier	AUC	Accuracy	Recall	Precision	F1
1	Yes	EMB	fastTextWiki	NB	0.80	0.76	0.78	0.75	0.76
2	Yes	Task	fastTextWiki	TT	0.81	0.76	0.71	0.79	0.75
3	Yes	EMB	IndicFT	NB	0.77	0.74	0.70	0.76	0.73
4	Yes	EMB	IndicFT	LR	0.78	0.74	0.70	0.75	0.73
5	Yes	EMB	IndicFT	SVC	0.79	0.74	0.71	0.76	0.73
6	Yes	EMB	IndicFT	ADB	0.79	0.74	0.72	0.76	0.74
7	Yes	EMB	IndicFT	XGB	0.79	0.74	0.70	0.76	0.73
8	Yes	EMB	Combined	NB	0.79	0.74	0.76	0.74	0.75
9	Yes	Task	mBERT (Pytorch)	PyrotchTT	0.80	0.74	0.69	0.76	0.72
10	Yes	EMB	fastTextWiki	SVC	0.81	0.74	0.67	0.79	0.72
11	Yes	Task	IndicFT	TT	0.81	0.74	0.71	0.76	0.74
12	Yes	EMB	IndicFT	DT	0.71	0.72	0.61	0.77	0.68
13	Yes	EMB	IndicFT	Perceptron	0.72	0.72	0.56	0.81	0.66
14	Yes	EMB	Combined	GBC	0.78	0.72	0.66	0.74	0.70
15	Yes	EMB	IndicFT	LGBM	0.79	0.72	0.67	0.74	0.70
16	Yes	EMB	IndicFT	GBC	0.79	0.72	0.67	0.75	0.71
17	Yes	EMB	IndicFT	RFC	0.79	0.72	0.68	0.75	0.71
18	Yes	EMB	fastTextWiki	LR	0.81	0.72	0.66	0.75	0.70
19	Yes	EMB	fastTextWiki	XGB	0.78	0.71	0.64	0.74	0.69
20	Yes	EMB	fastTextWiki	RFC	0.79	0.71	0.65	0.74	0.69
21	Yes	EMB	Combined	XGB	0.80	0.71	0.65	0.74	0.69
22	Yes	EMB	IndicBERT	RFC	0.71	0.70	0.70	0.70	0.70
23	Yes	EMB	fastTextWiki	LGBM	0.78	0.70	0.63	0.72	0.67
24	Yes	EMB	Combined	LGBM	0.78	0.70	0.63	0.72	0.67
25	Yes	EMB	Combined	ADB	0.78	0.70	0.64	0.74	0.68
26	Yes	EMB	fastTextWiki	ADB	0.79	0.70	0.65	0.73	0.69
27	Yes	EMB	Combined	LR	0.80	0.70	0.61	0.74	0.67
28	Yes	EMB	Combined	RFC	0.80	0.70	0.63	0.74	0.68
29	Yes	EMB	fastTextWiki	GBC	0.78	0.69	0.62	0.72	0.67
30	No		Keras_Tokenizer	CNN	0.74	0.68	0.65	0.70	0.67
31	No		Keras_Tokenizer	RNN	0.74	0.68	0.70	0.67	0.68
32	Yes	EMB	Combined	SVC	0.75	0.68	0.70	0.67	0.68
33	No		Lexical	LGBM	0.69	0.66	0.70	0.64	0.67
34	Yes	EMB	IndicBERT	XGB	0.71	0.66	0.65	0.67	0.66
35	No		Lexical	GBC	0.71	0.66	0.71	0.65	0.68
36	Yes	EMB	IndicFT	CNN	0.71	0.66	0.65	0.66	0.66

37	No		Lexical	SVC	0.72	0.66	0.69	0.66	0.67
38	No		Lexical	RFC	0.72	0.66	0.75	0.64	0.69
39	No		Lexical	LR	0.74	0.66	0.57	0.70	0.63
40	Yes	EMB	fastTextWiki	CNN	0.74	0.65	0.74	0.63	0.68
41	No		fastText	XGB	0.66	0.64	0.61	0.64	0.63
42	No		Word2Vec	LGBM	0.68	0.64	0.72	0.62	0.66
43	No		BOW	RFC	0.68	0.64	0.68	0.62	0.65
44	Yes	EMB	IndicBERT	LGBM	0.69	0.64	0.58	0.67	0.62
45	Yes	EMB	fastTextWiki	Perceptron	0.63	0.63	0.36	0.78	0.49
46	Yes	EMB	Combined	DT	0.63	0.63	0.63	0.63	0.63
47	Yes	EMB	fastTextWiki	DT	0.64	0.63	0.63	0.63	0.63
48	No		BOW	ADB	0.61	0.62	0.61	0.62	0.62
49	No		TFIDF	GBC	0.63	0.62	0.57	0.64	0.60
50	Yes	EMB	mBERT	DT	0.63	0.62	0.65	0.61	0.63
51	No		fastText	GBC	0.63	0.62	0.74	0.59	0.66
52	No		Word2Vec	ADB	0.64	0.62	0.71	0.61	0.65
53	No		Word2Vec	GBC	0.64	0.62	0.70	0.61	0.65
54	No		BOW	LGBM	0.65	0.62	0.56	0.63	0.59
55	No		BOW	GBC	0.65	0.62	0.55	0.63	0.59
56	No		fastText	LGBM	0.66	0.62	0.65	0.61	0.63
57	No		Word2Vec	SVC	0.67	0.62	0.75	0.60	0.66
58	No		TFIDF	SVC	0.68	0.62	0.58	0.64	0.61
59	Yes	EMB	IndicBERT	GBC	0.68	0.62	0.58	0.63	0.60
60	No		Lexical	ADB	0.68	0.62	0.63	0.62	0.63
61	No		Word2Vec	XGB	0.69	0.62	0.63	0.62	0.63
62	No		BOW	SVC	0.69	0.62	0.61	0.63	0.62
63	No		BOW	XGB	0.69	0.62	0.59	0.62	0.61
64	No		Word2Vec	LR	0.64	0.61	0.76	0.58	0.66
65	No		Lexical	DT	0.64	0.61	0.60	0.61	0.61
66	No		fastText	SVC	0.67	0.61	0.75	0.59	0.66
67	No		TFIDF	Perceptron	0.60	0.60	0.61	0.60	0.60
68	Yes	EMB	IndicBERT	DT	0.62	0.60	0.62	0.60	0.61
69	No		BOW	LR	0.63	0.60	0.50	0.62	0.56
70	Yes	EMB	mBERT	SVC	0.63	0.60	0.66	0.59	0.63
71	No		TFIDF	LR	0.64	0.60	0.52	0.61	0.56
72	Yes	EMB	IndicBERT	ADB	0.64	0.60	0.59	0.60	0.59
73	Yes	EMB	mBERT	GBC	0.64	0.60	0.65	0.60	0.62
74	Yes	EMB	mBERT	RFC	0.64	0.60	0.65	0.59	0.62
75	No		Lexical	XGB	0.64	0.60	0.63	0.59	0.61
76	No		TFIDF	LGBM	0.66	0.60	0.54	0.62	0.58
77	No		TFIDF	RFC	0.66	0.60	0.56	0.61	0.58
78	Yes	EMB	IndicBERT	NB	0.61	0.59	0.67	0.58	0.62
79	No		Word2Vec	DT	0.60	0.58	0.79	0.56	0.65

80	Yes	Task	mBERT (Transformer)	TransformerTT	0.60	0.58	0.65	0.57	0.61
81	No		TFIDF	DT	0.61	0.58	0.73	0.57	0.64
82	Yes	EMB	mBERT	LR	0.61	0.58	0.68	0.57	0.62
83	Yes	Task	IndicBERT (Transformer)	TT	0.61	0.58	0.63	0.57	0.60
84	Yes	EMB	mBERT	LGBM	0.63	0.58	0.64	0.57	0.60
85	No		fastText	LR	0.65	0.58	0.83	0.55	0.66
86	No		Lexical	NB	0.69	0.58	0.32	0.67	0.43
87	Yes	EMB	IndicBERT	LR	0.59	0.57	0.61	0.57	0.59
88	Yes	EMB	mBERT	XGB	0.61	0.57	0.62	0.56	0.59
89	No		Word2Vec	RFC	0.69	0.57	0.89	0.54	0.67
90	Yes	EMB	IndicBERT	Perceptron	0.56	0.56	0.22	0.67	0.33
91	No		fastText	Perceptron	0.56	0.56	0.65	0.56	0.60
92	No		TFIDF	NB	0.60	0.56	0.53	0.56	0.54
93	No		TFIDF	ADB	0.60	0.56	0.55	0.56	0.56
94	Yes	EMB	IndicBERT	SVC	0.60	0.56	0.65	0.55	0.59
95	No		fastText	ADB	0.61	0.56	0.52	0.56	0.54
96	No		TFIDF	XGB	0.65	0.56	0.59	0.56	0.58
97	No		fastText	RFC	0.71	0.56	0.90	0.54	0.67
98	No		BOW	Perceptron	0.55	0.55	0.51	0.55	0.53
99	No		fastText	DT	0.54	0.54	0.87	0.52	0.65
100	Yes	EMB	mBERT	ADB	0.56	0.53	0.67	0.52	0.59
101	No		Word2Vec	Perceptron	0.52	0.52	0.47	0.52	0.49
102	Yes	EMB	mBERT	NB	0.55	0.52	0.30	0.53	0.38
103	No		BOW	DT	0.56	0.52	0.51	0.53	0.52
104	No		BOW	NB	0.58	0.52	0.39	0.52	0.45
105	Yes	EMB	mBERT	Perceptron	0.50	0.50	0.24	0.51	0.33
106	No		Lexical	Perceptron	0.50	0.50	0.00	0.00	0.00
107	Yes	EMB	Combined	Perceptron	0.50	0.50	0.01	0.50	0.02
108	No		Word2Vec	NB	0.64	0.50	0.95	0.50	0.66
109	No		fastText	NB	0.66	0.50	0.95	0.50	0.66

APPENDIX E: RESEARCH PROPOSAL

1. Introduction / Overview

Mobile phones came to India in 1995[23] and Internet was launched in India by VSNL in 1995[24]. Initially the cost of the technology was extremely high, so it was available only to business class, research labs, high level bureaucrats and politicians. With the increase of literacy and decreasing cost of internet services and mobile phone device internet, it is so common that people started thinking that Internet is our fundamental right. As per the World Economic Forum (WEF), in 2019, about 60% of Indian internet users viewed content in vernacular. WEF also says 75% of this 60% is below 35 years of age (Wikipage, 2020b). According to the same Wikipedia page, by 2030, 1.1 billion Indian will have access to Internet and 80% will access the content on mobile devices. The WEF also estimated that 80% of the users will be consuming content in vernacular languages.

When Government of India is going for full blown Digital India program and bringing every citizen of India on the internet platform for purchase, payment and government fund transfer then how the citizens are going to provide feedback about the services which they use? As of today, it is easier to perform sentiment analysis of the feedback given in English but feedback given in Hindi is not easy to analyse. It means voice of Hindi speaking people is not being considered in service improvement. Till the time somebody is not too angry and do some crime or come on the road to do Dharana or protest we do not know what is happening and why.

Many Hindi new portals, book, blogs, chat bot/WhatsApp conversations, YouTube channels, Twitter & Facebook pages are full of content in Hindi language. People openly express themselves online using Hinglish language which is mix of Hindi, English, Urdu and other languages. Volume of the online content is increasing at unprecedented rate and it is responsibility of government, business community, professionals, NGO and others to understand the feeling of public and

[23] https://en.wikipedia.org/wiki/Telecommunications_in_India#:~:text=In%20August%201995%2C%20then%20Chief,launched%20in%20Kolkata%20in%202012. (Accessed 24-Jun-20)

[24] https://en.wikipedia.org/wiki/Internet_in_India#:~:text=The%20first%20publicly%20available%20internet,not%20permitted%20in%20the%20sector. (Accessed 24-Jun-20)

respond accordingly. But the biggest challenge is how to analyse the content which is written in mix of Indian languages. It is impossible to analyse the Hinglish language text manually or using traditional systems.

This section is organized as 1.1 What is Hinglish, 1.2 Origin of Hinglish, 1.3. What is Sarcasm?, 1.4. Why Sarcasm Detection is Critical?, 1.5. Why Sarcasm Detection is Critical in Electronic Media? , 1.6. Sarcasm Detection in Hindi, 1.7. Challenge in Processing Hinglish, 1.8. Common Challenges in Sarcasm Detection, 1.9. Context Understanding a Challenge in Sarcasm Detection, 1.10. Challenges in Sarcasm Detection in Hinglish, 1.11. Degree of Sarcasm, 1.12. Positive Side of Hinglish

1.1 What is Hinglish?

There was time when Hindi was a language which is used by majority of Hindi speaking people when they are communicating (writing, speaking) with each other. But in 21st century, most of the Hindi speaking population who express themselves on social media use Hinglish language. Hinglish is a new lingo of Hindi speaking population. Hinglish sentences follow Hindi grammar and most of the word are taken from Hindi but there is no hesitation of taking words from other languages like English, Urdu etc. Hinglish language spoken by different people have different amount of words from different languages. For example, those people who know Urdu good enough for them Hinglish is mix of Hindi, Urdu, English. Those who know Avadhi for them Hinglish is mix of Hindi, Avadhi, English. Those who know Marathi very well for them Hinglish is mix of Hindi, Marathi, English. Thus, in Hinglish Language we have words from Hindi, English and various other Indian languages and written in Devanagari & Roman together.[25] (Sinha and Thakur, 2005) Hindi and English language mixed is called Hinglish. Hinglish is not limited to Hindi & English mix but it includes Punjabi, Gujarati, Marathi, Urdu. Phrase construct happens in Roman and Devanagari script.[26]

[25] Latin is Region and Rome is part of that reason. Over the period of time Roman empire become famous and script was called Roman but Latin is also used simultaneously. https://www.quora.com/Why-is-the-language-of-the-ancient-Romans-called-Latin-and-not-Roman (Accessed 28-Jun-20)

[26] https://en.wikipedia.org/wiki/Hinglish (Accessed 24-Jun-20)

1.2 Origin of Hinglish

Before Internet Era in India people use to communicate with each other in much cleaner format of the language and there was not much mix of other language or English and for writing Hindi they were using Devanagari script. But, with the penetration of internet in the society a new language started taking shape. Initially when Devanagari keyboards were not available people were using Roman letters to write Hindi email, SMS.

An example of late 20th century text in Hinglish language. “Main is doorbhash ka prayog karna nani janta”. This is Hindi in Roman script. We need to keep in mind that people do not follow any IAST or other map for writing Hinglish letters in Roman. Mobile phone and Internet were available to elite, educated journalist, professionals. They started realising they are typing in Roman but some words in English so translating them and then typing in Roman is painful. So, text became like this “Main is phone ko use karna nahi janta”. Roman script with Hindi and English words.

Over the period of time when Devanagari keyboards were easily available people started using Devanagari keyboards for writing Hindi, but by that time so much English has come in day to day conversation that they felt it is uncomfortable to use Hindi words. So, they write like this.
“मैं इस फोन को यूज करना नहीं जानता”. Devanagari script with Hindi and English words. Over the period of time people started realizing it is becoming difficult to know which word is Hindi and which one is English therefore a word which come from English root should be written in Roman and word which are from Hindi root should be written in Devanagari. So, they started writing like this. “मैं इस phone को use करना नहीं जानता”. Devanagari & Roman mixed for Hindi and English words.

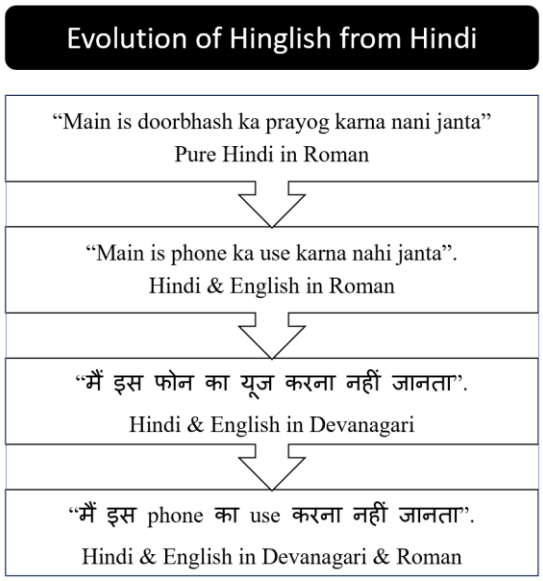


Figure 1: Evolution of Hinglish

Today if you read any Hindi speaker's WhatsApp, twitter or Facebook message you will find they use words from different Indian languages like Urdu, Marathi, Bangla, Punjabi and write either in Devanagari or in Roman. "अमी मोंजुलिका. अमी राजा को जरूर मारबो 😉, but why you want to kill him?". Here Hindi, Bangla, Urdu and English 4 languages used along with emoticon and written in two scripts Devanagari and Roman. This is Hinglish.

Today Hindi social media, Hindi comment boxes of product, Hindi news articles are full of this kind of language, Hinglish. Therefore, this work using Hinglish language is high value from the angle of practical usage.

1.3 What is Sarcasm?

Your friend come to you and speak something to you, from the tone of his language, his body language, choice of his words, time and situation he is speaking you realised that the real meaning of what he is saying is completely opposite. It may be easier for you to detect this opposite sense if you are aware about the complete context but if you are not aware about the context then even as intelligent human you may miss the real meaning of what is being said.

For example, you open the door for your friend, and he says wow! your looking handsome in this T-shirt. You know that this is an old T-shirt and many times your friend has seen this. But still not aware of full context, you hesitantly say thank and you invite him inside. After 15 minutes you check yourself in the mirror and realised that you are wearing T-shirt flip side. Now you are embarrassed for your "Thank you" response.

What your friend did was sarcastic remark on your dressing and you being unaware of the full context could not respond properly. In the absence of full context, understanding sarcasm is difficult task and most of the time we take literal meaning of the words or some other time get confused that why someone has made that remarks which was completely out of the context.
In English language this type of grammatical construct which has completely opposite meaning than what is said, it called sarcasm.

As per merriam-webster dictionary, sarcasm is[27]
1: a sharp and often satirical or ironic utterance designed to cut or give pain
2a: a mode of satirical wit depending for its effect on bitter, caustic, and often ironic language that is usually directed against an individual
2b: the use or language of sarcasm

In Hindi it has several name and synonyms like कटाक्ष (Kataksha), तंज (Tanja), व्यंग/ व्यङ्ग (Vyanga), टोंट (Tonta)

Ten forms of humour are irony, satire, sarcasm, overstatement, self-deprecation, teasing, replies to rhetorical question, clever replies to serious statements, and transformations of frozen expressions. All these are functions of humour and found in the sitcom (situational comedy). What one finds hilarious or pun may be completely opposite to another person in another country or in other situation. Interpretation is filtered by cultural context. (Anggraini, 2014)

[27] https://www.merriam-webster.com/dictionary/sarcasm

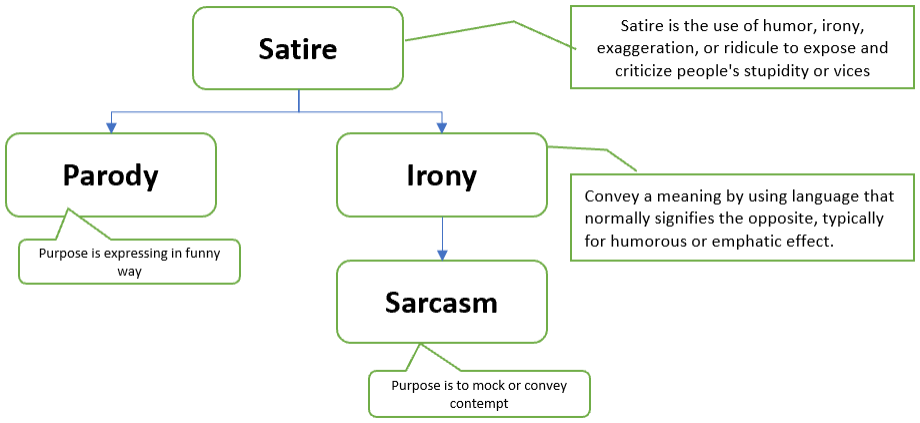


Figure 2: Sarcasm & Satire Relationship

In their work "The Differential Role of Ridicule in Sarcasm and Irony" (Lee et al., 2009) says sarcasm and irony are similar because they are both form of reminder yet they are different because sarcasm is about ridiculing a specific person however this is not required in case of irony. However, in our work we will ignore this specific aspect. There are two reasons for that 1- we are interested in predicting whether the statement is conveying real meaning or opposite meaning of what is being said 2- In Indian and specific to Hinglish context words like कटाक्ष or व्यंग doesn't consider the aspect mentioned by the authors.

1.4 Why Sarcasm Detection is Critical?

If we do not understand the real intent of the speaker then we cannot respond him properly. Response can be physical action or verbally reply to the speaker or even no action.

Few examples where not understanding the real intent of the person can be catastrophic.

- In face to face communication with your customer when you miss his intent. Result is customer disengagement.
- In live program when you are listening a response or question from the audience in hall or live TV or Radio program or speaking over phone or video conferencing tool and you miss the intent. Result is dent on your reputation.
- In offline communication when you publish some content on blog, news, product selling page and receive some comment from the public. Someone expresses his opinion over your

post or tweet and you are not able to understand that properly or not able to read. All other people read that comment and think that either you are dumb or do not care or accept what is being said. Result you know very well.

When you are dealing with your known people, friends, relatives and not responding properly in that situation, it will have lessor impact because they know your real nature and potential. But in public places, where you do not know the person to whom you need to respond, can cause huge dent on your image and brand.

1.5 Why Sarcasm Detection is Critical in Electronic Media?

With the advancement of online sales of products, social media and online blogs, new portals there is huge surge of online feedback. Post COVID19 pandemic there are clear trends of shifting in this direction. People prefer buying, reading, expressing, engaging online. This justifies the need of sophisticated real time sarcasm detection system.

1.6 Sarcasm Detection in Hindi

English is 3rd most spoken language in the world and many researchers across the world are working for sarcasm detection in English. But, Hindi is 4th most spoken language in the world and not much significant work is happening in sarcasm detection in Hindi. Due to this reason many of the feedback given on twitter, Facebook, product page, online news goes unnoticed.

Sarcasm is one kind of feedback and if we do not use this to improve our response then we prove ourselves foolish and customer shift to different product, service or platform. Similar things happen when people change their party or group. Therefore, we feel it is extremely important to detect the sarcastic feedback given by those people who write in Hindi.

1.7 Challenge in Processing Hinglish

A. Complexity due to English words in Hindi

Observe the variation of a sentence “I have purchased tickets”

मैनें (टिकिटें/ टिकटें/ टिकटे/ टिकिट) खरीद (ली/ लीं) (है/हैं). This simple sentence can be spoken in 16 different ways if written in Devanagari. If we mix Roman script in between then number

of permutations goes beyond our normal imagination. Here me need to make note that Ticket is English word, and people are making plural of that as they do with any Hindi word.

Let us see another sentence “She has boiled the rice”

उसने राइस बोइल कर दिया है

From the above Hinglish sentence you cannot figure out whether the doer is female or male. Secondly, राइस and बोइल are not words in Hindi dictionary. Sometime people will write letter in Roman like

उसने Rice बोइल कर दिया है / उसने Rice Boil कर दिया है / उसने राइस Boil कर दिया है /

उसने Rice बोयल कर दिया है

Like Guru, Karma are Hindi words and they are part of English dictionary. We do not have Hinglish dictionary which has word like यूज गुड नाइस क्वीन in that dictionary. Without transliterating words like Tickets, Boil into Devanagari and telling system that टिकिटें = टिकटें = टिकटे= टिकिट, बोइल= बोयल embedding will not give good results.

B. Mix Other Indian Language with Hindi

Observe the sentence below, Bangla written in Devanagari and clearly understandable by any Hindi speaking person. Most of the words in the sentence below are from Bangla language but written in Devanagari.

अमी मोंजुलिका.अमी राजा को मारबो दीदी ने केजरीवाल को भी पीछे छोड़ दिया. जि तो कमालई कर दओ ददद्दू

India’s business film Industry in Mumbai make film in Hindi. Rarely any film use as good Hindi as Hollywood uses English. Adoption of words from other language is not a problem. The problem is quantity of the words taken from other languages, availability of the updated vocabulary of the language. Many famous dialogues or songs from Hindi films

which are taken different language or dialects. This increases complexity of sarcasm detection in Hinglish. We do not have comprehensive dictionary which we can call Hinglish dictionary which has all the word being used by the Hinglish speakers.

Without telling system that अमी (Bangla word) = मैं, मारोबो (Bangla word) = मारूंगी = मारूंगा = मारना no embedding is going to help

C. Complexity of Synonyms in Hindi

For this let's understand what Synonyms is. A word or phrase that means exactly or nearly the same as another word or phrase in the same language[28], for example "shut" is a synonym of "close". Few examples of synonyms

- The East = The Soviet Union (https://www.lexico.com/en/definition/synonym)
- Country of rising sun = Japan, Dragon Country = China,
- Fridge = Refrigerator
- Happy = Joyful, Cheerful, Contented, Jolly, Gleeful, Carefree

All the synonyms have different spelling, different pronunciation but almost same meaning and part of the same language. l'eau (French word for water) is not synonyms of water because they are two different languages.

Unlike other world languages, all Indian languages (except Tamil, this is debatable) heavily borrow words from Sanskrit.

Let's take English word "Water" and see how many words are available in sanskrit for "water" जल = पानी = तनि = नीरू = आपः = वाः = वारि = सलिलं = पयः = तोयं = मेघपुष्पं = घनरसः = पाणी. So all these words are synonyms of water in sanskrit.

Because all Indian languages have root in Sanskrit therefore most of the time, they take word from Sanskrit for communication. For example, Kannada uses नीरू, Bangla use पानी,

[28] https://www.lexico.com/en/definition/synonym.

Hindi uses पानी, सलिलं, मेघपुष्पं. If not regular, they are used in poetical or sometimes in sarcastic language. Because in sarcasm or poetry we often use loaded words.

In Hindi language, can we say नीरू is synonym of पानी? No, because नीरू word is normally is used in Kannada and Sanskrit and not in Hindi. As per the definition of synonym another equal word should be from the same language and we know Hindi is not Kannada nor it is Sanksrit. The answer is yes also; because Sanskrit being mother of Hindi language, it borrows words freely from Sanskrit. Thus, we see synonym in Hinglish is not the way it is understood in the context of English.

Therefore, to be build a complete Hinglish dictionary we have take words from all other Indian languages and frequently used English words as well. Thus it should be like this.

जल = पानी = तनि = नीरू = आपः = वाः = वारि = सलिलं = पयः = तोयं = मेघपुष्पं = घनरसः = वाटर

D. Variation in Spelling of Same Word

In Hindi same word spoken and written with different spelling. Observe the spelling of the same word how they are varying. This kind of problem we do not have in English. As discussed earlier, synonym of Happy is Jolly. They both are not same, neither in spelling, nor in pronunciation, nor in full sense, but "happy" is close to "jolly". That is why they are synonyms. But below all "=" signs are referring to the same thing.

विष्णु = बिश्णु = विश्णु = बिष्णु = विष्नु = बिष्नु,

दरसन= दर्शन= दर्सन = दरशन

करता = कर्ता,

यज्ञ = जग्य,

योग = जोग,

हरि=हरी,

We need to keep in mind Hindi is not Devanagari, nor Hindi is Avadhi or Marathi. Hindi is written in Devanagari script but it is heavily inflicted by other languages like Awadhi, Bhojpuri, Rajasthani, Urdu etc.

Unless we have dictionary which tells विष्णु = बिश्णु = विश्णु = बिष्णु = विष्नु = बिष्नु, no embedding will help.

1.8 Common Challenges in Sarcasm Detection

Detecting Sarcasm is difficult if sentences are having following characteristics.

E. **Idioms and Phrases**: Sarcasm detection become more difficult when people speak in idiomatic language. For example: What a wise man! what he did is nothing other than an axe to grind.

F. **Speaking with Hint:** When people do not talk directly and use examples which are completely different than context. For example: You are behaving like Mir Jafar.

G. **Culture:** Different languages have different degree of challenges in sarcasm detection. For example, English is spoken all over the world but the way American express their feeling is different than the way British express. The reason for that is the work and social culture of England and United States is hugely different. In English language what is call sarcasm in England may be considered a normal statement or abusive in US and vice versa.

1.9 Context Understanding a Challenge in Sarcasm Detection

Since the time human child take birth, baby has environment to learn from. Various types of formal or informal environment, social or business or cultural background forces human to think and learn. Either at physical or emotional or intellectual level if human fail to learn then his survival is challenged by the nature around. In this kind of environment, it is easy for any human to understand the context. If we are alert and interested in the topic then we need not to struggle hard to understand the context. But context understanding is extremely difficult in the case of Machine learning. Let us analyze one sarcastic tweet. "#JIO का सच नीता अंबानी ने मन्नत मांगी थी कि अनंत अम्बानी अपना वजन कम कर लेगा तो गरीबों में 3 महीने Net or call का भंडारा करवाऊँगी"

People living in India can understand that this is sarcasm. Because we know the full context. That

- Mukesh Ambani is owner of #Jio

- Neeta Ambani is Wife of Mukesh Ambani
- Anant Ambani is son of Neeta Ambani
- Anant Ambani has 200+ Kg body weight
- Normal body weight of human is around 70 kg
- Anand Ambani is overweight as per the normal standard
- Neeta Ambani desired that her son should have normal weight
- #Jio has launched 3 Month free internet package
- There is no direct connection between Anand Ambani weight reduction and 3-month free internet package

(Joshi et al., 2018) in their work "Investigation on Computational Sarcasm" says there are three type of context, Author Specific context, Conversational Context, Topical Context

We need to understand that keeping all the facts in mind we can say a statement is sarcasm and not normal statement. Even a human, who does not have all this information will fail to classify a statement as sarcasm. It is not easy to give all this information to a system to make a classification decision

1.10 Challenges in Sarcasm Detection in Hinglish

A. 70% of the world population uses 26 letters of Roman script to write their language. The Roman alphabet is also used as the basis for the International Phonetic Alphabet, which is used to express the phonetics of all languages.[29] Due to this reason when people are writing different language like English, French, Indonesian, Tagalog, German, Turkish they need not to change much around the letters, so most of the cases script remain Roman. This advantage is not available to Devanagari script and Hindi language.
B. An average westerner knows and speaks one language so written and verbal expression most of the time is that one language. An average Indian speaks minimum 2 languages, one is language of his state, plus national language, or English. In southern part of India, it is not uncommon when you find a taxi or truck driver who can speaker 3 or 4 languages but they cannot speak in English. This, one language- one script, advantage is not available

[29] https://www.worldatlas.com/articles/the-world-s-most-popular-writing-scripts.html Accessed on 23-Jun-20

for any Indian and they communicate in multiple language without realising that they have shifted language and borrowing words from different language.

C. While typing feedback people write @account_name. Most of the time @account_name are proper name and written in Roman like @harithapliyal, @eating_point, @banarasi. Similarly, hashtag, which helps us understanding the context of the feedback, is also written in Roman script #Election2019 #COVID19 #Philosopy #Motivation #NarendraModi.

D. Numerals: Many times, people use non English numerals like १, २, ३, ४, ५.

1.11 Degree of sarcasm

Although how a person perceive & responds to a sarcasm it also depends upon him, yet we need to know all sarcastic statements are not equally intense or powerful to generate pain to the listener or reader. Here are few examples of different degree of sarcasm.

E. ओ भाई कचोरी समोसे की दुकानें खुल तो गयी है लेकिन ध्यान रखे कचोरी समोसे के चक्कर में आप की ही पूड़ी सब्जी न बट जाये #Covid_Unlock (Least Intense)

F. NDTV की हैडलाइन एक बेजुबान अल्पसंख्यक भैंस को डूबा कर मारने की कोशिश करती बहुसंख्यक चिड़िया (Lessor intensity)

G. करोना का दवा न होना यह एक साइंस है, और दवा न होते हुए भी बिल लाखों मे आना ये एक आर्ट है !! (Moderate Intensity)

H. ये शुक्र है जंगल में आरक्षण नहीं, बहोत नहीं तो जंगल का राजा शेर नहीं गधा होता. आरक्षण खत्म करो 70 साल हो गये यार #आरक्षण_भीख_है (Sharp Intensity)

1.12 Positive Side of Hinglish

Although India is big country with 1.35 billion people with different culture, religion, tradition but there is some common aspect in India culture and this does not change no matter where a Indian is living on the earth. That common culture helps us understanding the context and intent easily. Although there are many languages in India but because of one overarching culture it is easier to understand the meaning, a simple translation is good enough. Unlike English where Australian struggle to understand what American gentlemen want to say in English.

2. Background and Related Work

(Bharti et al., 2017) Sarcasm detection is one of the most complex work in Hindi Language and the reason for that is words in Hindi language are rich in morphology. This paper discusses a system to sarcasm detection in Hindi tweet but for that it is taking help of online news related to the tweet. This work demonstrates accuracy of 70.4%

Let us take one English verb "do", in Hindi, it can be used like कर्ता (noun) , करता (verb with male), करती (verb with female), करूंगा (future tense with male), करूंगी (future tense with female), किया (done), करो (must do) करें (please do) etc. these all are with different gender, mood and tenses. However, in English we have infliction like do, does, did, done.

Now, let's take another example but this time we take noun "Ram". राम का, राम ने, राम को, राम द्वारा, राम में, राम पर, राम के लिए राम पर and many times you will see letters are written together. We never see any word like "ByRam" in English but in Hindi रामने and राम ने both have same meaning.

Sarcasm is the major factor which can flip the meaning of a written or spoken phrase. To avoid the negativity people use positive words to communicate negative message. (Desai and Dave, 2016). They have used libsvm algorithm for multiclass classification. This paper uses 5 grades of sarcasm Non-Sarcastic, Mild Positive Sarcastic, Extreme Positive Sarcastic, Mild Negative Sarcastic and Extreme Negative Sarcastic. This work demostrate the accuracy between 60% to 84% depending upon, whether sentence has any clue like emoticon, tag etc of sarcasm. This work suggests usage of lexical, pragmatic, and lingustic features along with emoticons, hashtag, punctuation marks to detect the sarcasm.

(Liebrecht et al., 2013) developed a sarcasm detection system. This was system was developed for tweets in Dutch language. They used 78,000 sarcastic tweets, along with normal tweets dataset, while adding normal tweet ensured that none of the normal tweet is part of sarcastic dataset. Split the sarcastic tweets into train-test and added with normal tweet into train dataset to train the model. Then test the model using test dataset which has only sarcastic tweets. There experiments leads to

AUC of .79. This paper gives an overall approach of building sarcasm detection system in other than English language. But it does not address the problem which Hinglish language has. There test train split and model training approach looks good for non-English language.

(Asghar et al., 2014) developed a system to detect negative, positive, and neutral sentiments for English language tweets. As claimed by the authors their system can detect and score the slang used in the tweet. This system has Accuracy of 92% for binary classification and 87% for multinomial classification. An approach to get tweets clean text is discussed for English language tweets. However, we need to look what extra we need to do for Hinglish language tweets.

(Turney, 2002) presents an unsupervised learning-based algorithm for classification of review in English language. Semantic Orientation (SO) is used to perform this work. SO of a phrase is calculated using adverbs and adjectives used in the phrase. The experiments were done for text of various domains like automobiles, banks, movie review and travels. The results of this experiment vary from domain to domain between 66 to 84%. The power of this SO in Hinglish language sarcasm detection can used and verified.

Lot of work has been in English language sarcasm detection and authors mentioned different challenges in sarcasm detection, although results are not that great as for any other classification problems. Challenges exists because of context understanding, missing context, domain, culture, different words, or expression used by people to flip the meaning etc. There is not much work done in Hinglish Language Sarcasm detection. Hinglish language has a separate set of challenge like mixing script, mixing language, highly morphological words, using same morphology on English language words, meagre size of corpus etc.

3. Research Questions

E. How sarcasm detection is done by other researchers for English and any other Indian languages?
F. How sarcasm detections system should be designed when words from more than one scripts are used for communication. For example: “मेरा work पूरा हो गया है”, it has 2 scripts.

G. How sarcasm detection system works when more than language are used for communicating idea. For example: “मेरा वर्क पूरा हो गया है”, it has 2 languages.

H. Unlike English, Hindi is highly morphological language how does it influences overall approach? For example, करता, करती, करते all are equivalent to English “do” but depends upon the gender.

I. Unlike Roman where we write words using consonants and vowels in Devanagari there is an extra concept called Maatra, this is not available in Roman script. For example, word “Experience” in Roman is written using 5 vowels, 5 consonants. But in Devanagari it is written as “एक्सपिरिऐन्स” 2 vowels (ए ऐ), 6 consonants(क् स् प् र् न् स्) , 5 Maatra (ा, ि, ि, े, ा). How does Maatra of Devanagari influences text processing?

J. How to do transliteration from Roman to Devanagari? Many options are available for reverse translation. For example “एकीकरण” => “Ekikaran” is easy lot many option there but “Ekikaran” => “एकीकरण” is not easy. Because Hindi speaking population is not aware about IAST[30] and nor they use it for transliteration. So confusion is “ra”=> र or ऱ, n=> न or न् or ण or ण् or ञ or ञ् or ङ or ङ्, ki=> कि or की or क्ि or क्ी or क् इ or क् ई

K. What kind of feature engineering need to be done when text is in multiple scripts?

L. When some English word is written in Devanagari i.e. राइस, कुक then how to handle these words because they are not part of normal Hindi dictionary?

M. Can we use the same approach for other Indian language words in Devanagari i.e. बरमंड (Garhwali word for “brain”), तुस्सी (Punjabi word for “you”), खाबो (Bangala word for “eating”)?

N. If we create a feature using hashtag and say feature name is “context” then is it good enough to explain the context and produce better result?

O. Can NER based features help in sarcasm detection?

4. Aim and Objectives

The aim of this research is to propose a model, which can predict sarcasm in a given Hinglish language sentence with highest possible accuracy.

[30] https://en.wikipedia.org/wiki/International_Alphabet_of_Sanskrit_Transliteration

Based on the above primary goal, objectives of this research are as following.

E. To analyze the existing dataset of 300+ statements, clean it and labels each sentence.
F. To expand existing dataset, minimum 600%, which can be used for training and testing a sarcasm detection model of Hinglish Language
G. To determine which embedding technique best suits for Hinglish dataset
H. To develop a preprocessing pipeline which can handle Hinglish language sentences.
I. To develop models using different algorithm like Naive Bayesian, SVM, Logistic Regression, Recurrent Neural Network, and other. Consider minimum 4 suitable algorithms.
J. To develop a prediction model using ensemble of different model and check whether performance improves.
K. To evaluate different models and identify the best model.

To address issue related to the small dataset set we will use cross validation technique. Because we are going to develop this dataset therefore, we will try to create a balance dataset and hence no oversampling technique will be required. But, if we realize that results are not encouraging and we need to expend our dataset then in the interest of time we will put more non-sarcasm sentences and use oversampling technique to balance the dataset.

5. Significance of the Study

We didn't find one place which has done research and can say with conviction that approximately these are the number of Hindi speaker in the world. Different sources reveal different numbers. As per a lingoda.com[31] and babbel.com[32] after English and Mandarin Hindi is 3rd most spoken language on earth. It is spoken by 615mn people. As per Wikipedia 176 million people speak Urdu.[33]

[31] https://blog.lingoda.com/en/most-spoken-languages-in-the-world-in-2020 Accessed on 22-Jun-20

[32] https://www.babbel.com/en/magazine/the-10-most-spoken-languages-in-the-world Accessed on 22-Jun-20

[33] https://en.wikipedia.org/wiki/Urdu Accessed on 22-Jun-20

Culture of Hindi speaking population and Urdu speaking population resembles a lot. While speaking or writing Hinglish many words of Urdu are spoken or written unknowingly. Therefore, any sarcasm analysis system in Hinglish will benefit Urdu speaking community as well.

With current trend of increasing online content in Hindi, it is practically not possible to read each and every review, even if you try it is very expensive and not worth work. We know, even one negative feedback or abuse which goes unnoticed can cause huge problem for the brand of the company, product, or person. Therefore, performing sentiment analysis on every feedback makes a perfect sense and it can be done automatically almost in real time.

Sarcasm is one type of sentiment and we are trying to discuss overall benefits of sentiment analysis keeping Sarcasm at the centre of discussion.

E. Sentiment analysis has a broad range of applications like understanding whether a feedback is Sarcasm, Warning, Love Emotion, Hate Emotion, Advertisement of some other product, Contradicting statement, Pun, Abuse, Inspiring Quote, Sensational Revelation, Pleasant Surprise, Allegation, Poetry/Dohe/Chands etc.
F. Government, NGO, religious leaders, product sellers are able to perform the sarcasm analysis against some product, political party, ideology, religion, company etc then they will be able to control the situation in much better way with minimum damage.
G. Sarcasm analysis can be used to analyse the feedback on airlines service, travel service, bus or taxi service, telecom, health, government service, new articles, personal blog, food delivery, insurance service, personality page, book page are good places where sentiment analysis plays a critical role.
H. In multinational companies it becomes exceedingly difficult to use humour to communicate the idea, crack joke or sarcasm, even if all the team member can speak English. The reason for that is different cultural background and different level of comprehension of English by non-native speakers. But when Hindi speaking people connect over video, telephonic or chat conversation it is easy for them to use idioms, joke, sarcasm and ensure that idea is understood. There is different kind of joy of working in lesser formal and light-hearted environment. When India people are speaking to each other using Hinglish we can perform sarcasm analysis to know the feeling of the group.

We are writing the examples of motivation in English language so that we can explain how sarcasm detection can help proper response from chatbot, but common use-case remain same.

Motivation in Travel Domain

Passenger: #ac_not_working. I love to get roasted in heat.

Chatbot: Sorry for the inconvenience. Our service engineer will call you.

Motivation in Hospital Business

Attendant: #expensive_treatment. We come to your hospital for this expensive treatment so that we can talk to your cute nurses.

Chatbot: We understand your concern about treatment cost. Our billing manager will call you.

Motivation in Restaurant Business

Customer: Last time, your food was so good that since last 2 days I am taking rest.

Chabot: I am sorry to hear that.

Motivation in Learning Portal

Learner: What a great content. I am still trying to understand the head and tell of that 30 min video.

Chatbot: Sorry, can you please share with us what difficulty you faced ?

Motivation in News Portal

Reader: What a great story! Did you read it after writing?

Chatbot: We are sorry that you didn't like this story.

Motivation in Airlines Business

Traveler: First time in my life I got such a wonderful service from any airlines. I reached to the destination one day before my check-in baggage.

Chatbot: We are sorry to hear that. We hope your baggage reached safe to you.

Motivation in Dialogue Analysis Work

A dialogue from a Hindi Film “Sholey”[34]

मौसी मेरा दोस्त इतना अच्छा है कि वह शराब को कभी न नहीं बोल पाता। पीने के बाद जुआ खेलना उसकी खूबी है इसमें उसका कोई दोष थोडी है मौसी। बस हारने के बाद थोडा मारपीट करता है और घर में आ के मेरे को गाली देता है। पर मेरा दोस्त दिल का बहुत अच्छा है मौसी आप अपनी बेटी की शादी मेरे दोस्त से पक्की कर दो

This is a pure sarcasm paragraph. These kind of dialogues makes movie interesting.

6. Scope of the Study

G. This research is not related to any specific domain like philosophy, politics, history, current affair new etc. Rather it is trying to detect sarcasm in day to day informal conversation.

H. Sarcasm in our communication can be expressed and experienced at Visual (facial express, body language), Vocal (tone, pace of speech, emphasis on certain word) and text (book, newspaper, articles, social media tweets, comments and feedback box on internet. Visual sarcasm is more universal than vocal. Because voice uses language and there are 7000+ languages on the earth so there is no universal vocal language of expressing sarcasm. But pause, pitch, pace, modulation between words, while speaking, are more universal like Visual. In this paper we are deal only with text-based sarcasm.

I. Only Roman and Devanagari scripts are considered.

J. Only Hindi and English language words are considered. If heavily used words from other languages which are part of day to social communication, then we will include that in our Hindi vocabulary.

K. No analysis of degree of sarcasm.

L. We know to understand the context datetime plays a critical role. And most of the text in the dataset is coming from tweet. Our base dataset does not have datetime. We could have included datetime. But we avoided that intentionally because in future when we are expending the dataset further, we will extract information from different books and other sources and that time datetime will not be available. We wanted to develop a generic system

[34] https://en.wikipedia.org/wiki/Sholay

which can understand the context using hashtag. Hashtag is part of the tweet. And we will be extracting it as a separate feature. We do not want that our system should be depending upon time to understand the context.

7. Research Methodology

In this section we are going to discuss a high-level approach to accomplish the research goal. The flow of discussion in the section is as following 7.1. About Dataset, 7.2. Dataset Structure, 7.3. Handling Small Dataset size, 7.4. Building Dataset, 7.5. Cleaning Text, 7.6. Labelling, 7.7. Transliteration, 7.8. Context Creation, 7.9. Emoticon Handling, 7.10. Embedding, 7.11. Feature Engineering, 7.12. Algorithm, 7.13. Prediction, 7.14. Result Comparison

7.1 About Dataset

Keeping the duration of project is mind, it was recommended that we should use an already existing dataset. We used Hindi tweet dataset.[35] This excel fine had total 442. During the project planning phase we realised that these 442 are not sarcastic tweet but mix of normal and sarcastic and to determine the sarcastic-ness of a sentence developer of this dataset is using news context and there is not explicit labelling available in the given dataset.

Based on the feedback from research guide we decided to expand the dataset which should have minimum 1000 sarcastic sentences and 1000 normal sentences. To develop a dataset with minimum 2000 sentences we had adopted following approach.

1. Clean the base file and label the tweets as sarcastic and normal.
2. Updated dataset will also have non-tweet sentences
3. These 2000 sentences will be marked as sarcasm or normal by a team of minimum 3 people
4. Finally, whatever is the maximum vote will be the label of the sentence

7.2 Dataset Structure

5. Dataset will have 3 columns “Sentence”, “Context”, “Label”
6. Sentence: Sentence is text of the tweet or any normal sentence.

[35] https://github.com/rkp768/hindi-pos-tagger/tree/master/News%20and%20tweets (Accessed on 26-Jun-20)

7. Context: This will be written in the hashtag format (one word). Those tweets which has hashtag it can be extracted from the text and for non-hash tagged sentences and non-tweet sentences context, it will be created manually. Many times, sentence will not have any context. For example “हां मुझे गाली सुनना बहुत पसन्द है” “Yes, I love to hear abuses” This is a sarcastic sentence and there is no context required.
8. Label: This column will have 0 for normal sentence and 1 for sarcastic sentence.

7.3 Handling Small Dataset size

We will build our dataset which has 1000 sarcastic statements and 1000 non-sarcastic statements. Because dataset is not large enough therefore, we will use cross validation of 5 folds. For developing neural network-based model we will use 10 folds oversampling.

7.4 Building Dataset

Identify some twitter accounts, hashtags which posts sarcastic text. Write some code in python using tweepy to extract the text from these hashtags and accounts. Extract text from some blogs which write sarcastic articles. Extract each sentence of the blog as a record. Save all these tweets and sentences from the blog into a csv file.

7.5 Cleaning Text

We know that tweet text is unclean because it has text from different languages, in different scripts, extra space, emoticons, non-text sign like "~" ":", "<" etc, flag sign, line break, over used words like ".....", "??????", "beau.....tiful", "!!!!!!". Blog text may also have this kind of text but chances of that is extremely less. We will write a python script to clean all records. Now onwards we will not refer this as tweet or blog text but as sentences. Save all the clean sentence text in a new csv file.

7.6 Labelling

Identify 3 or 5 good Hindi reader who can read the text and identify which sentence is sarcasm and which not. Every manual labeller will label the sentence independently. After getting input from all the people majority of vote will decide whether a sentence is sarcastic or not.

7.7 Transliteration

We know Roman typing is much easy compare to typing in Devanagari therefore many time people use Roman letters in between the sentence. This is true especially if it is name of politician, film actor, place name, (#AmitShah, #Modi, #Salman, #Khan, #India #Bollywood, #Delhi, #Karnataka #Yogi) etc. Because same word will be written in Devanagari and other times in Roman and this is not good for text analysis. So, we will transliterate all the Roman words into Devanagari.

7.8 Context Creation

Whether a sentence is sarcastic or normal sentence, it also depends upon context. For example, "Thank you so much for your help" is normal sentence. But if context is "BJP said to Rahul Gandhi after winning election" then earlier sentence is sarcastic. We will use hashtag of the tweets to extract the context. If tweet has more than one hashtags then we will combine them using "_". If there is no hashtag, which will be true if text is taken from blog, in that we will manually write context. Context will not be sentence but one or two words connected with "_". We want to understand if context is given as hashtag and not as full sentence then how does it impact sarcasm detection.

7.9 Emoticon Handling

We will create another feature called "Emotions" using emoticons found in tweet. We will use corresponding English language word for creating this feature. Text taken from blog will not have any emoticons.

7.10 Embedding

(Sharma et al., 2014) in their work “A Sentiment Analyzer for Hindi Using Hindi Senti Lexicon” suggests using bootstrap approach to extract senti words from Hindi Wordnet. It has given encouraging results of 87% accuracy in sentiment analysis. We are going to test usefulness of this approach in sarcasm detection.

In their paper, Adaptive GloVe and FastText Model for Hindi Word Embeddings, (Gaikwad and Haribhakta, 2020) states that AGM gives better results than GloVe and FastTextWeb. They also mentioned that FastText embeddings which are trained on FastTextHin (Hindi Monolingual

corpus) produce better results than FastTextWeb. We are planning to use FastTextHin corpus to check the performance.

Google research has introduced a multilingual BERT which is capable of working with more than 100 languages (Romano, 2020). We will use this for our project and check how it can be used and how it performs for the task of sarcasm detection in Hinglish.

7.11. Feature Engineering

We will explore different methods for creating feature. For example:

e) Based on number of Adjective or Adverbs
f) Hashtag,
g) Emoticon
h) Bag of bowls using one word, two words, three words

In their work, (Joshi et al., 2018) have used 3 types of features POS, Named Entities, Unigram to predict the disagreement. However, the results are not encouraging but we would like to explore these features for sarcasm detection.

7.12. Algorithm

Depending upon time available and performance we can include more algorithm, but we will use following 4 algorithms to develop models.

a. SVM, b. Logistic Regression, c. RNN/GRU/LSTM, d. Naïve Bayesian

7.13 Prediction

Models developed with different embedding and algorithm with be used to predict the result on test dataset. We will use train-test split of 50:50 and 80:20 and check which split helps training the model better.

7.14 Result Comparison

The result of prediction will be compared using Recall, Precision, Accuracy & F1-Score. Results of best models will be ensembled and best possible result with ensembled model will be discussed.

This will step will helps us knowing that which embedding and algorithm works the best for Hinglish Language sarcasm detection.

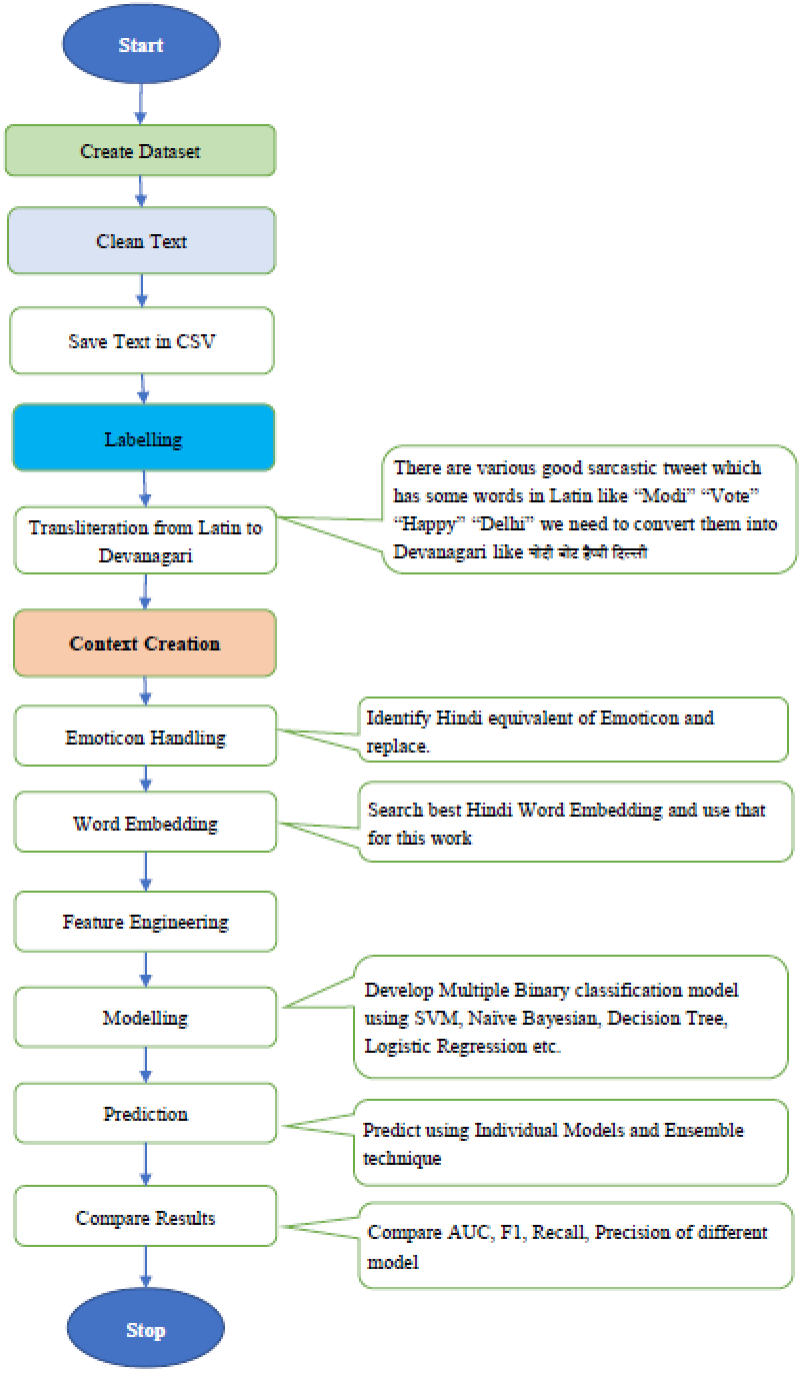


Figure 3: Overall Approach

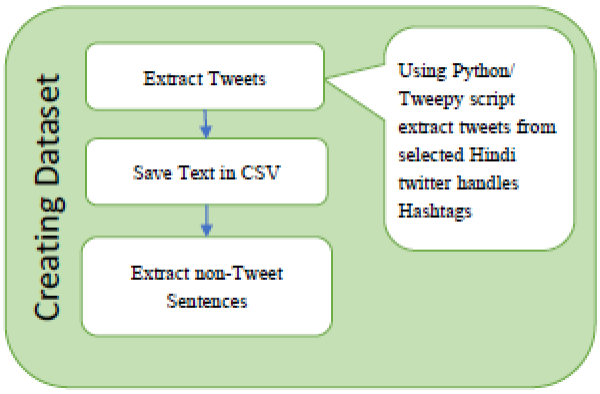


Figure 4: Steps to Creating Dataset

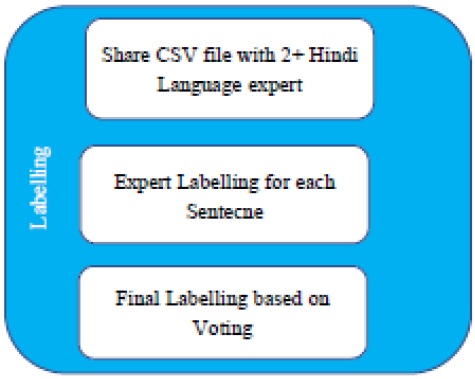


Figure 5: Steps for Labelling Sentences

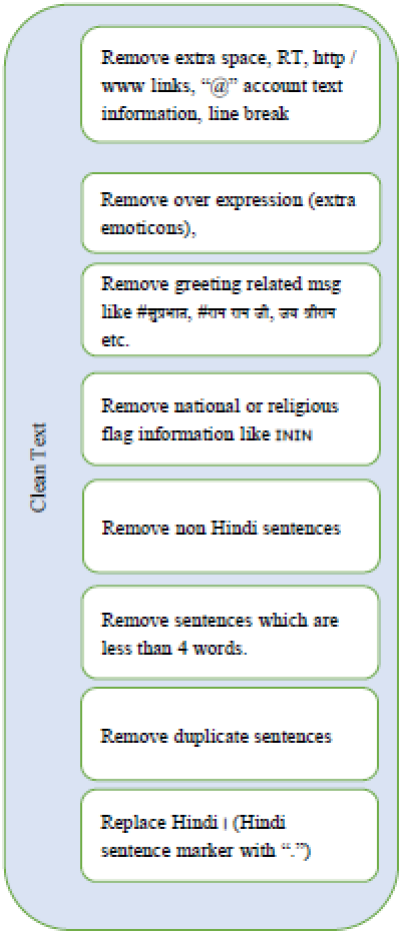


Figure 6: Steps to Create Clean Sentences

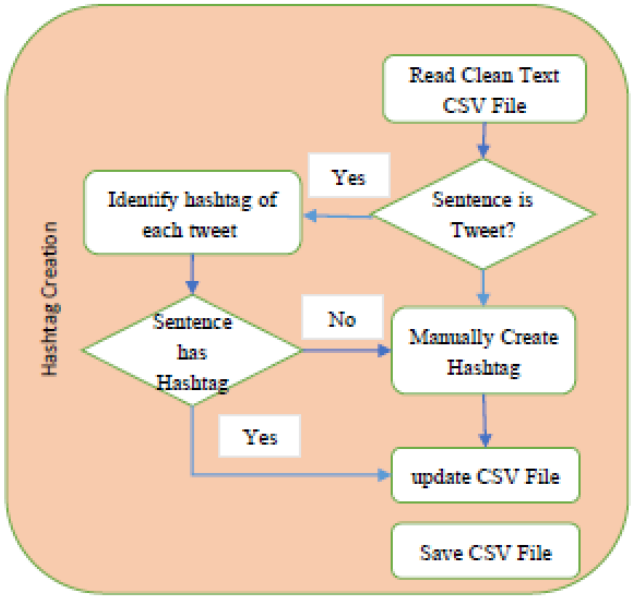


Figure 7: Steps to Create Hashtags

Evaluation Metrics

ROC graphs are useful tool for visualizing and evaluating classifiers. ROC are able to provide a richer measure of performance than accuracy or error rate (Fawcett, 2004). However, for sake of illustration we will also use Accuracy, F1, Recall & Precision, because they have their relevance depending upon the domain where we use this for sarcasm detection. For example "Hospital administrators thinks I come to hospital because I have lot of money they have beautiful nurses to chat with" (writing sarcasm in English to make sure more readers understand the impact of choice of evaluation metrics). Healthcare domain, hospital administrators may be taking any sarcasm seriously and they do not want any sarcasm to be misclassified and they are ready for more False-

True. To illustrate the choice of metrics, lets assume there are 1000 sentences in the dataset, 150 are sarcasm and 850 are normal sentences. Let's say Model1 predicts 110 are sarcasm and 890 normal and Model2 predicts 140 sarcasm and 860 normal sentences. Accuracy of both the models is 90%. If we select Recall and F1 score then Model1 is better. If we select precision then Model2 is better. If we need to detect sarcasm in comment box of YouTube channel of some political party then we can go for Model1 which is giving recall of 73%. If we are dealing with some more serious product or service like healthcare, airlines service then we can go for Model2 which is giving Precision score of 63%.

Model1

		Observation		
		FALSE	TRUE	
Actual	FALSE	820	30	850
	TRUE	70	80	150
		890	110	1000

Accuracy	0.90
Recall	0.73
Precesion	0.53
F1 Score	0.81
Error Rate	0.10

Model2

		Observation		
		FALSE	TRUE	
Actual	FALSE	805	45	850
	TRUE	55	95	150
		860	140	1000

Accuracy	0.90
Recall	0.68
Precesion	0.63
F1 Score	0.77
Error Rate	0.10

Figure 8: Model Selection based on Evaluation Metrics

8. Expected Outcomes

a) Tagged dataset of 2000 sentences
b) A system to detect the sarcasm.
c) Best practices for feature creation in Hinglish language NLP work

9. Requirements / resources

Hardware

a) Laptop (already have)

Software/Packages

a) Multilingual BERT
b) Google Colab (available)
c) NLTK (available)

d) scikit-learn.org (available)
e) seaborn (available)
f) matplotlib (available)
g) Google Sheet (for creating dataset)
h) Microsoft Word (available)
i) Mendeley (available)
j) Hindi SentiWordnet
k) Indic Translation

10. Research Plan

10.1 Risks or contingency plan

10. Risk #	11. Risk Name & Response Plan
12. 1	13. **Risk**: Latin to Devanagari Transliteration May be more complex than planned 14. **Contingency Plan**: If we are not able to find or build a suitable solution for translation then we will proceed without transliteration or perform manual transliteration.
15. 2	16. **Risk:** Due to non-availability of any good corpus of Named Entities in Hindi we may not be able to perform NER tagging of sentences. 17. **Contingency Plan:** We will drop NER experiment from this project.
18. 3	19. **Risk:** If time is constrained and we may not able to write context of all then sentences 20. **Contingency Plan:** We will develop two solution a- with only those sentences which context b- without context column. Whatever gives better results we will make conclusion based on that.
21. 4	22. **Risk (Positive Risk):** If we have more time and primary goal is achieved. 23. **Contingency Plan:** We will increase dataset size and perform experiments on the new dataset.

10.2 Project Schedule

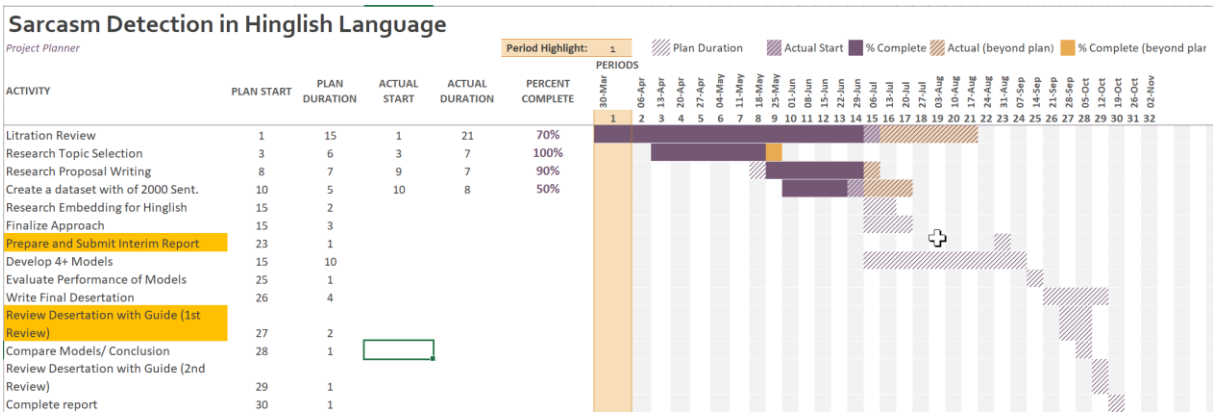


Figure 9: Project Schedule