A Survey on Emotion Detection from Text

Abhishek Mahakal[1*†]

[1*]Computer Science, Pennsylvania State University, 777 W Harrisburg Pike, Harrisburg, 17057, Pennsylvania, USA.

Corresponding author(s). E-mail(s): amm10344@psu.edu;
[†]These authors contributed equally to this work.

Abstract

This survey paper provides a comprehensive review of research on emotion detection from text over the past decade. The paper is organized into four sections, each covering a specific time period: before 2010, 2010-2014, 2014-2018, and 2018-2023. The first section discusses early approaches to emotion detection, which were primarily rule-based and focused on identifying emotional keywords or phrases. The second section covers the use of machine learning techniques for emotion detection, including feature-based approaches, which use handcrafted features, and deep learning approaches, which learn features automatically from data. The third section discusses the growing importance of context in emotion detection, including the use of sentiment analysis and topic modeling techniques to provide additional context for emotion classification. The final section covers the latest developments in the field, including the use of multimodal data sources and the integration of affective computing and natural language processing. Throughout the paper, code snippets are provided to illustrate the implementation of key concepts and techniques discussed in each section. Overall, this survey paper provides a valuable resource for researchers and practitioners interested in emotion detection from text, highlighting key trends, challenges, and future directions for the field.

Keywords: Emotion detection, rule-based, deep learning, sentiment analysis, multimodal data sources

1 Introduction

The recognition of emotions has been a topic of interest for several decades, with researchers developing techniques to detect emotions through various physiological characteristics such as voice, facial expressions, hand gestures, or body movements. Emotions can also be detected using speech and image signals, with some researchers combining different emotion recognition models to achieve better results.

However, there has been little research on recognizing emotions from only textual input, even though natural language plays an important role in emotion recognition. Traditionally, research in this area has focused on the discovery and utilization of emotional keywords, which are specific words that express the speaker's emotional state. While using emotional keywords is the most direct way to recognize a user's emotions from text input, it has its limitations, such as ambiguity in defining all emotional keywords, recognizing emotions from sentences with no emotional keywords, and the lack of semantic and syntactic information for emotion recognition.

Some researchers have used other textual information clues such as pragmatic intent, text content plausibility, and paragraph structure to recognize emotions from the text. Litman and Forbes integrated all dialog information, including acoustic and linguistic features, dialog acts, and the sequence of speakers, to recognize the emotional state in a dialog system. Schuller et al. integrated both acoustic and linguistic information in emotion recognition.

With further analysis, some researchers believe that textual data is rich with emotion at the semantic level, that is, that emotional content is also contained in the semantic structure. Chan and Franklin analyzed input sentences and constructed a symbolic network to reduce language model perplexity. Woods used a transition network to analyze natural language. Chuang and Wu proposed a semantic network-based emotion recognition mechanism using emotional keywords, semantic/syntactic information, and emotional history to recognize the emotional state of a speaker.

Research in emotion recognition has focused on discerning emotions along the dimensions of valence (positive/negative) and arousal (calm/excited) and on recognizing distinct emotion categories. Liu et al. used a real-world commonsense knowledge base to classify sentences into Ekman's basic emotion categories. They used an ensemble of rule-based affect models to determine the emotional affinity of individual sentences. Neviarouskaya et al. also used rules to determine the emotions in sentences in blog posts, relying on a manually prepared database of words, abbreviations, and emoticons labeled with emotion categories.

Although communication systems can identify the users' emotional states from different communication modalities, the variety and complexity of language makes it difficult for researchers to recognize emotional states from pure textual data. Recognizing emotion is extremely important for some text-based communication tools, such as the dialog system, which is a kind of human-machine communication system that uses only text input and output. Recognizing the users' emotional states enables the dialog system to change the response and answer types. Text is still the main communication tool on the Internet.

In conclusion, emotion recognition is an important area of research, and while significant progress has been made in recognizing emotions through physiological characteristics and speech and image signals, recognizing emotions from only textual input remains a challenge. While keyword-based approaches have been used traditionally, they have their limitations, and researchers are exploring other textual information clues and semantic networks to improve emotion recognition. Recognizing emotions from text is important for various communication systems, including dialog systems, and will continue to be an area of interest for researchers in the future.

2 Summary of papers before 2010

2.1 Emotion Recognition from Text Using Semantic Labels and Separable Mixture Models

The paper proposes a new approach to emotion recognition from text using semantic labels and separable mixture models. The authors argue that traditional approaches to emotion recognition often rely on pre-defined sets of emotion categories and do not capture the complexity and variability of emotional experiences. Instead,[1] they propose a method that is able to identify the specific emotions expressed in a piece of text by modeling the underlying semantic structure of the language.

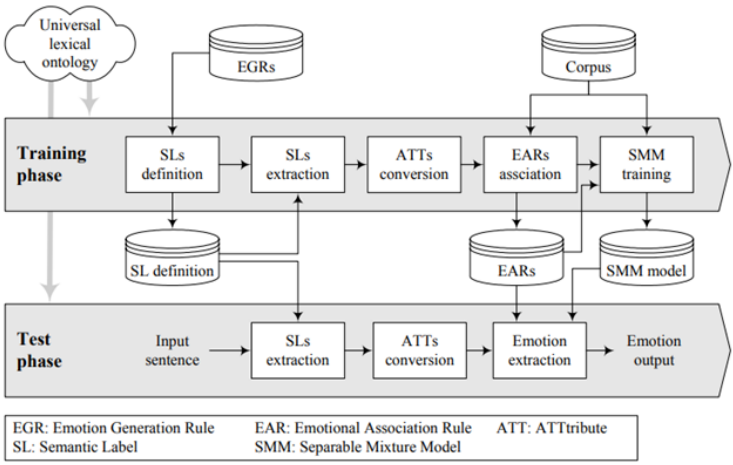

Figure 1: System Block Diagram

The proposed approach consists of two main components: semantic labeling and separable mixture models. In the first component,[1] the authors use a pre-trained semantic parser to identify the syntactic and semantic structure of the input text. The parser breaks down the text into its constituent parts (e.g. subject, verb, object) and assigns semantic labels to each part based on its function in the sentence (e.g. agent, patient, instrument).

Once the input text has been labeled semantically, the authors use a separable mixture model to identify the specific emotions expressed in the text.[1] A separable mixture model is a probabilistic model that can be used to represent the joint distribution of multiple variables. In this case, the variables are the semantic labels assigned to the input text and the emotions expressed in the text. The model is "separable" because it allows the authors to model the relationship between the semantic labels and the emotions separately.

To train the separable mixture model, the authors first create a training dataset consisting of text samples annotated with the specific emotions expressed in the text. They then use a clustering algorithm to group similar semantic labels together, which reduces the number of variables in the model and makes it more computationally efficient. Finally, they use a maximum likelihood estimation algorithm to fit the model to the training data.

The experimental results in the paper were divided into four experiments:

Experiment 1 aimed to determine the best parameter combination for the entire system. The results showed that the optimal parameter combination for the proposed approach was a combination of semantic labels and separable mixture models.

Experiment 2 evaluated the accuracy of emotion recognition using the proposed approach. The experimental results showed that the proposed approach outperformed other approaches, such as the maximum entropy approach and the SVM approach.

Experiment 3 tested the generalization ability of the proposed approach. The system was ported into another domain, which was a dialogue system for customer service. The experimental results showed that the proposed approach maintained high accuracy in recognizing emotions in a different domain.

Experiment 4 compared the proposed approach with other approaches in the literature. The experimental results showed that the proposed approach achieved higher accuracy in recognizing emotions than other approaches.

Two dialogue corpora were collected for the experiments.[1] Corpus A consisted of 26 volunteer college students who recorded dialogs about their daily life. The data was collected over one month, and the students were asked questions about their daily life. They were divided into two groups, and the emotions of the students in Group 1 were recorded for 40 dialogs for each emotion, while for the remaining 20 students, 10 dialogs were recorded for each emotion.

Corpus B contained the dialogs from a broadcast drama. The emotions of the performers in the drama were manually tagged with one of the three emotional states: happy, unhappy, and neutral. The dialogs of the protagonists were preserved in the corpus, and sentences with three words or less were discarded.

In both corpora, the sentences were annotated with one of the three emotional states, and sentences that did not belong to these three emotional states were deleted from the corpus. The experimental results showed that the proposed approach achieved high accuracy in recognizing emotions in both corpora.

Overall, the experimental results showed that the proposed approach, which used semantic labels and separable mixture models, achieved high accuracy in recognizing emotions in text, even when applied to a different domain. The results also showed that the proposed approach outperformed other approaches in the literature.

2.2 Emotion Recognition From Textual Input Using An Emotional Semantic Network

The paper focuses on the problem of recognizing emotions from textual input, which is an important task in natural language processing. Emotion recognition is challenging because emotions are often expressed implicitly in text, and it can be difficult to capture the relationships between words and emotions.

To address this problem, the authors [2] propose an approach that involves constructing an emotional semantic network from a large corpus of text. The network maps words to emotions and captures the relationships between emotions. The authors then use the network to classify text into different emotion categories.

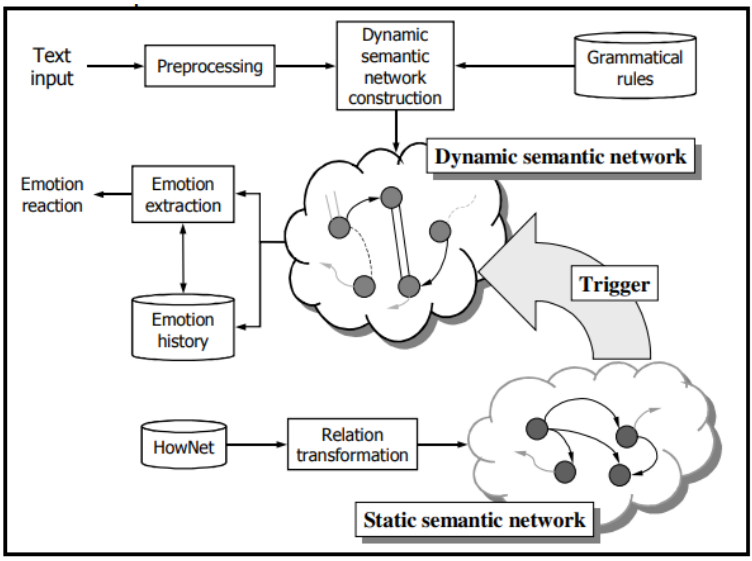

Figure 2: Block Diagram of the emotion recognition system

The authors begin by collecting a large corpus of Chinese texts and extracting the emotional content of the texts using a set of emotion keywords. They then use this data to construct an emotional semantic network. The network consists of nodes that represent words and emotions. They are basically the emotion carrier. There are 5 components in a node: Word Name, trigger value acceptor, initial emotion vector, emotion propagation value, emotion vector. And edges that represent the relationships between them. There are 4 types of edges: subject to object, subject to indirect object, bi-directional link between subject and complement, and equal link between subject and object. The authors use a clustering algorithm to group emotions together and to capture the relationships between them.

Once the emotional semantic network is constructed, the authors use it to classify text into different emotion categories. The classification process involves mapping words in the text to nodes in the network and propagating the emotions through the network to obtain a final classification.[2] The authors use a probabilistic approach to propagate the emotions through the network, taking into account the strength of the relationships between the emotions. The propagation of emotions is triggered by a new input sentence and ends with the emotion propagation values below a threshold.

If concept C is defined by k definitions (D_1 to D_k) with k relationships (r_1 to r_k), shown as:

$DEF = r_1 D_1, r_2 D_2, \ldots, r_k D_k \quad (1)$
[2] The emotion trigger value T_c of a concept C is calculated as:

$$T_c = \frac{1}{k} \sum_{i=1}^{k} F(r_i, D_{T_i}) \quad (2)$$

The decay function is calculated as:

$$E_P = D(E_{t-1}) = \begin{cases} E_{t-1} & \text{, equal link} \\ \delta(l) \times W(E_{t-1} + 0.5g \times T_C) & \text{, other wise} \end{cases}$$

$$\delta(l) = \exp(-0.05 \times l^2)$$

$$W(x) = \begin{cases} 1, & -1 \le x \le 1 \\ 0, & \text{other wise} \end{cases}$$

Figure 3: Equations for Decay Function calculation
The authors evaluated their approach on a dataset on a broadcast drama that consisted of lead male and lead female dialogues.[2] There were 558 sentences contained in 137 dialogues from the leading man and 453 sentences contained in 136 dialogues from the leading woman. According to the experimental results, using this proposed system, then emotion recognition of textual input achieved an accuracy of about 60%.

	Male			Female			Average
	Tag	Rec	Rate	Tag	Rec	Rate	
happiness	104	72	69.23%	121	80	66.12%	67.67%
anger	92	71	77.17%	80	63	78.75%	77.96%
surprise	84	54	64.29%	66	49	74.24%	69.26%
sadness	137	83	60.58%	92	56	60.87%	60.73%
fear	108	62	57.41%	63	35	55.56%	56.48%
antipathy	33	14	42.42%	31	7	22.58%	32.50%
total	558	356	63.80%	453	290	64.02%	63.91%

Figure 4: Emotion Recognition Result

The authors also discuss the potential applications of their approach in various domains, such as in sentiment analysis, customer feedback analysis, and opinion mining. They note that their approach could be applied to a wide range of text-based applications where emotion recognition is important.

In conclusion, the paper presents an approach for recognizing emotions from textual input using an emotional semantic network. The approach involves constructing a network that maps words to emotions and captures the relationships between emotions, and using the network to classify text into different emotion categories. The authors evaluate their approach on a dataset of Chinese texts and report promising results, and discuss the potential applications of their approach in various domains.

2.3 Using Roget's Thesaurus for Fine-grained Emotion Recognition

The paper proposes a method for fine-grained emotion recognition in text using Roget's Thesaurus. Emotion recognition is an important task in natural language processing and has many applications, such as sentiment analysis, opinion mining, and dialogue systems. However, existing approaches to emotion recognition typically rely on a limited set of predefined emotion categories, which may not capture the complexity and nuance of human emotions.

To address this limitation, the authors propose a method that leverages Roget's Thesaurus, which contains a comprehensive list of words and their semantic relationships. Specifically, the authors use a set of 20 emotion categories derived from Roget's Thesaurus, which cover a wide range of emotions such as joy, anger, fear, and sadness.

Roget's Thesaurus is a comprehensive list of English words and their semantic relationships. The authors use Roget's Thesaurus to [3] define a set of 20 emotion categories, each of which is associated with a set of words that

are indicative of that emotion. For example, the emotion category "joy" is associated with words such as "happy", "pleased", and "delighted", while the emotion category "anger" is associated with words such as "mad", "upset", and "irritated". By using Roget's Thesaurus to define these emotion categories, the authors are able to capture a wide range of emotions and their semantic relationships.

This was the best summer I have ever experienced.	*(happiness)*
I don't feel like I ever have that kind of privacy where I can talk to God and cry and figure things out.	*(sadness)*
Finally, I got fed up.	*(disgust)*
I can't believe she is finally here!	*(surprise)*

Figure 5: Sample Sentences from corpus

The authors first annotate a dataset of sentences with the 20 emotion categories using crowd-sourcing. They then extract lexical features from each sentence, such as unigrams, bigrams, and part-of-speech tags, and use these features to train a machine learning classifier. The classifier predicts the most likely emotion category for a given sentence based on its lexical features.

They used [3] a corpus of blog sentences annotated with emotion labels for their experiments. They trained classifiers with unigram features for each emotion class using Support Vector Machine (SVM) for predicting the emotion category of the sentences in their corpus. SVM has previously given good performance in sentiment classification experiments.

In their experiments, they used ten-fold cross-validation experiments conducted using the SMO implementation of SVM in Weka. In each experiment, they represented a sentence by a vector indicating the number of times each feature occurs. They conducted three experiments: (1) using only corpus-based unigram features, (2) using features from the emotion lexicon acquired from Roget's Thesaurus (RT), and (3) combining corpus-based unigrams with RT features and features from WordNet-Affect.

The results showed that the combination of corpus-based unigram features and features derived from emotion lexicons contributed to improved performance, better than given by any one type of feature group alone. They found that [3] the correlation between the features and the predicted class was highest for the "happy" class, indicated by a precision of 0.813 and recall of 0.698, the highest among all classes. This suggests that it is easier to discern happiness in text than Ekman's other basic emotions.

Model	Class	Precision	Recall	F-Measure	Baseline F-Measure
Unigrams	Happiness	**0.840**	0.675	0.740	0.469
	Sadness	**0.619**	0.301	0.405	0.368
	Anger	0.634	0.358	0.457	0.379
	Disgust	**0.772**	0.453	**0.571**	0.179
	Surprise	**0.813**	0.339	0.479	0.306
	Fear	**0.889**	0.487	0.629	0.506
	No-emotion	0.581	0.342	0.431	0.579
Roget's Thesaurus (RT) Features	Happiness	0.772	0.562	0.650	0.469
	Sadness	0.574	0.225	0.324	0.368
	Anger	0.638	0.246	0.355	0.379
	Disgust	0.729	0.297	0.421	0.179
	Surprise	0.778	0.243	0.371	0.306
	Fear	0.857	0.470	0.607	0.506
	No-emotion	0.498	0.258	0.340	0.579
Unigrams + RT Features	Happiness	0.809	**0.705**	**0.754**	0.469
	Sadness	0.577	0.370	**0.451**	0.368
	Anger	0.636	0.419	0.505	0.379
	Disgust	0.686	0.471	0.559	0.179
	Surprise	0.717	0.374	0.491	0.306
	Fear	0.831	0.513	0.634	0.506
	No-emotion	0.586	0.512	0.546	0.579
Unigrams + RT Features + WNA Features	Happiness	0.813	0.698	0.751	0.469
	Sadness	0.605	**0.416**	0.493	0.368
	Anger	**0.650**	**0.436**	**0.522**	0.379
	Disgust	0.672	**0.488**	0.566	0.179
	Surprise	0.723	**0.409**	**0.522**	0.306
	Fear	0.868	**0.513**	**0.645**	0.506
	No-emotion	**0.587**	**0.625**	**0.605**	0.579

* Highest precision, recall, and F-measure values for each class are shown in bold

Table 1: Emotion-related words automatically extracted from Roget's Thesaurus

In conclusion, the authors were able to demonstrate that a combination of corpus-based unigram features and features derived from emotion lexicons can help automatically distinguish basic emotion categories in written text. The effectiveness of their emotion lexicon derived from Roget's Thesaurus based on semantic relatedness of words to a set of basic emotion words for each emotion category was also demonstrated.

```
# Print the emotions associated with each word
# The sentence is: "The weather is calm"
for word in word_emotions:
    print(word, word_emotions[word])
    print("\n")
```

```
weather [('foul-weather_gear', 'foul-weather_gear'), ('weather_sheet', 'sheet'), ('weathercock', 'weathercock'), ('weather_dec
k', 'weather_deck'), ('weatherglass', 'weatherglass'), ('weather_map', 'weather_map'), ('weather_chart', 'weather_map'), ('weat
her_radar', 'weather_radar'), ('weather_satellite', 'weather_satellite'), ('weather_ship', 'weather_ship'), ('weather_strip',
'weather_strip'), ('weatherstrip', 'weather_strip'), ('weather_stripping', 'weather_strip'), ('weatherstripping', 'weather_stri
p'), ('weathervane', 'weathervane'), ('weather_vane', 'weathervane'), ('weather_eye', 'watchfulness'), ('weatherliness', 'weath
erliness'), ('weather_forecasting', 'meteorology'), ('weather_forecast', 'weather_forecast'), ('weather_outlook', 'weather_fore
cast'), ('weather_bureau', 'weather_bureau'), ('weather_station', 'meteorological_observation_post'), ('weatherman', 'weatherma
n'), ('weather_forecaster', 'weatherman'), ('cold_weather', 'cold_weather'), ('fair_weather', 'fair_weather'), ('hot_weather',
'hot_weather'), ('weather', 'weather'), ('weather_condition', 'weather'), ("poor_man's_weatherglass", 'scarlet_pimpernel'), ('w
eatherboard', 'to_windward'), ('weather_side', 'to_windward'), ('raw_weather', 'raw_weather'), ('weatherboard', 'clapboard'),
('weatherboarding', 'clapboard'), ('conditions', 'weather'), ('conditions', 'conditions'), ('conditions', 'conditions'), ('nois
e_conditions', 'noise_conditions'), ('ski_conditions', 'ski_conditions'), ('meteorological_conditions', 'meteorological_conditi
ons'), ('atmospheric_condition', 'weather'), ('endure', 'digest'), ('endure', 'suffer'), ('endure', 'survive'), ('endure', 'pre
vail'), ('endure', 'last'), ('endure', 'wear'), ('endure', 'weather'), ('outbrave', 'outbrave'), ('outbrave', 'outbrave'), ('br
ave', 'weather'), ('brave_out', 'weather'), ('upwind', 'upwind')]
```

```
is [('belch', 'burp'), ('behave', 'act'), ('bend_over_backwards', 'fall_over_backwards'), ('slumber', 'sleep'), ('bed_down', 'b
ed_down'), ('bed', 'go_to_bed'), ('unbend', 'unbend'), ('unbend', 'relax'), ('limber_up', 'limber_up'), ('beam', 'beam'), ('bel
lylaugh', 'bellylaugh'), ('barber', 'barber'), ('enrobe', 'enrobe'), ('bedizen', 'bedizen'), ('bear', 'wear'), ('disrobe', 'und
ress'), ('robe', 'vest'), ('beget', 'beget'), ('bear', 'give_birth'), ('bear', 'have_a_bun_in_the_oven'), ('blubber', 'snive
l'), ('beat', 'exhaust'), ('be_sick', 'vomit'), ('limber', 'limber'), ('slabber', 'drivel'), ('slobber', 'drivel'), ('rubberiz
e', 'rubberize'), ('rubberise', 'rubberize'), ('rubber', 'rubberize'), ('leave_behind', 'leave'), ('bear_upon', 'affect'), ('be
ar_on', 'affect'), ('conglobe', 'conglobate'), ('sober_up', 'sober_up'), ('sober', 'sober_up'), ('sober_up', 'sober_up'), ('sob
er', 'sober_up'), ('sober', 'sober'), ('become', 'become'), ('bedew', 'bedew'), ('bedraggle', 'bedraggle'), ('beef_up', 'streng
then'), ('besprinkle', 'sprinkle'), ('circumscribe', 'limit'), ('number', 'number'), ('benficiate', 'benficiate'), ('belly', 'b
elly'), ('belly_out', 'belly'), ('alphabetize', 'alphabetize'), ('alphabetise', 'alphabetize'), ('embellish', 'fancify'), ('bel
ch', 'erupt'), ('bedim', 'obscure'), ('benight', 'benight'), ('bedim', 'benight'), ('barbeque', 'barbeque'), ('barbecue', 'barb
eque'), ('befall', 'happen'), ('bechance', 'happen'), ('befall', 'befall'), ('bechance', 'befall'), ('betide', 'befall'), ('bes
tir_oneself', 'get_cracking'), ('leave_behind', 'leave'), ('come_into_being', 'come_to_life'), ('bestialize', 'bestialize'),
('bestialise', 'bestialize'), ('becalm', 'steady'), ('unbend', 'relax'), ('be_on', 'get_on'), ('belittle', 'diminish'), ('clabb
er', 'curdle'), ('befoul', 'foul'), ('reverberate', 'reverberate'), ('become_flat', 'die'), ('transcribe', 'transcribe'), ('ben
eficiate', 'beneficiate'), ('be_with_it', 'know_the_score'), ('be_on_the_ball', 'know_the_score'), ('imbibe', 'imbibe'), ('beat
_in', 'hammer_in'), ('bear_in_mind', 'mind'), ('remember', 'remember'), ('remember', 'commemorate'), ('remember', 'remember'),
('misremember', 'misremember'), ('be_amiss', 'misconstrue'), ('read_between_the_lines', 'read_between_the_lines'), ('befuddle',
'confuse'), ('bedevil', 'confuse'), ('beat', 'perplex'), ('bewilder', 'perplex'), ('subedit', 'copyread'), ('cube', 'cube'),
('beat_about', 'cast_about'), ('label', 'label'), ('label', 'label'), ('describe', 'identify'), ('bear', 'digest'), ('bear_up',
'bear_up'), ('rubberstamp', 'rubberstamp'), ('bethink', 'bethink'), ('believe', 'believe'), ('believe', 'believe'), ('misbeliev
e', 'misbelieve'), ('believe', 'think'), ('take_to_be', 'think_of'), ('be_after', 'plan'), ('draw_a_bead_on', 'draw_a_bead_o
n'), ('bet', 'count'), ('believe', 'believe'), ('beware', 'beware'), ('flabbergast', 'flabbergast'), ('ascribe', 'impute'), ('b
elieve_in', 'believe_in'), ('prescribe', 'order'), ('begin', 'begin'), ('bear_down', 'bear_down'), ('featherbed', 'featherbe
d'), ('bespeak', 'request'), ('beg', 'beg'), ('beseech', 'bid'), ('bear_on', 'push'), ('gibber', 'gibber'), ('bewitch', 'hex'),
('bewitch', 'magnetize'), ('besiege', 'besiege'), ('beg', 'solicit'), ('probe', 'probe'), ('beep', 'beep'), ('proscribe', 'forb
id'), ('subscribe', 'subscribe'), ('beg', 'beg'), ('deliberate', 'debate'), ('deliberate', 'consider'), ('beatify', 'beatify'),
('berate', 'call_on_the_carpet'), ('belabor', 'belabor'), ('belabour', 'belabor'), ('beat_around_the_bush', 'beat_around_the_bu
sh'), ('belie', 'misrepresent'), ('make_believe', 'make'), ('bear_down', 'bear_down'), ('betray', 'denounce'), ('belittle', 'di
sparage'), ('besmirch', 'defame'), ('libel', 'libel'), ('bemock', 'mock'), ('gibe', 'jeer'), ('betray', 'deceive'), ('befool',
'gull'), ('exuberate', 'exuberate'), ('belittle', 'minimize'), ('beshrew', 'curse'), ('bedamn', 'curse'), ('betoken', 'bode'),
('remember', 'commend'), ('remember', 'remember'), ('betroth', 'betroth'), ('inscribe', 'inscribe'), ('beg_off', 'excuse'), ('b
eef', 'gripe'), ('bellyache', 'gripe'), ('bewail', 'deplore'), ('bemoan', 'deplore'), ('bet', 'bet'), ('bespeak', 'bespeak'),
calm [('discomposure', 'discomfiture'), ('calming', 'appeasement'), ('calm', 'composure'), ('calmness', 'composure'), ('calm_ai
r', 'calm_air'), ('calm', 'calm_air'), ('calmness', 'calmness'), ('equanimity', 'composure'), ('calm_down', 'lull'), ('calm_dow
n', 'calm'), ('calm_down', 'calm'), ('be_quiet', 'close_up'), ('disquiet', 'perturb'), ('quiet', 'calm'), ('quieten', 'calm'),
('quiet', 'quieten'), ('quiet_down', 'quieten'), ('tranquilize', 'sedate'), ('tranquilize', 'calm'), ('tranquillize', 'sedat
e'), ('tranquillize', 'calm'), ('tranquillise', 'sedate'), ('tranquillise', 'calm'), ('lull', 'lull'), ('lull', 'lull'), ('lul
l', 'calm'), ('distill', 'distill'), ('distill', 'distill'), ('distill', 'distill'), ('distill', 'condense'), ('still', 'hus
h'), ('distill', 'purify'), ('instill', 'impregnate'), ('still', 'still'), ('instill', 'inculcate'), ('hold_still_for', 'stand_
for'), ('instill', 'impress'), ('still-hunt', 'still-hunt'), ('still-fish', 'still-fish'), ('instill', 'instill'), ('still', 'c
alm'), ('still', 'still'), ('instill', 'instill'), ('steady_down', 'settle'), ('steady', 'steady'), ('steady', 'brace'), ('go_s
teady', 'go_steady'), ('becalm', 'steady'), ('cool_off', 'cool'), ('cool_off', 'calm'), ('cool_off', 'cool_off'), ('chill_out',
'calm'), ('simmer_down', 'calm'), ('settle_down', 'settle'), ('settle_down', 'calm'), ('settle_down', 'settle'), ('cool_it', 'c
alm'), ('unagitated', 'calm'), ('serene', 'serene'), ('serene', 'calm'), ('tranquil', 'placid'), ('tranquil', 'calm'), ('tranqu
ilizing', 'ataractic'), ('tranquillizing', 'ataractic'), ('tranquilising', 'ataractic'), ('tranquillising', 'ataractic')]
```

Code 1: Code Output

3 Summary of papers between 2010 & 2014

3.1 Unsupervised Emotion Detection from Text Using Semantic and Syntactic Relations

The paper proposes a novel unsupervised context-based emotion detection method that does not rely on any affect dictionaries or annotated training data, unlike the traditional methods that depend on supervised learning methods that has a large set of annotated data. But the availability of such large annotated data is very rare and a model trained on a particular domain is not easily ported to another domain. The authors work with a small set of representative words which are used to compute an emotion vector of an affect-bearing word by calculating the semantic relatedness score between this word and an emotion concept. After this, fine-tuning is carried out based on the context of the word. This approach shows promising results in the end.

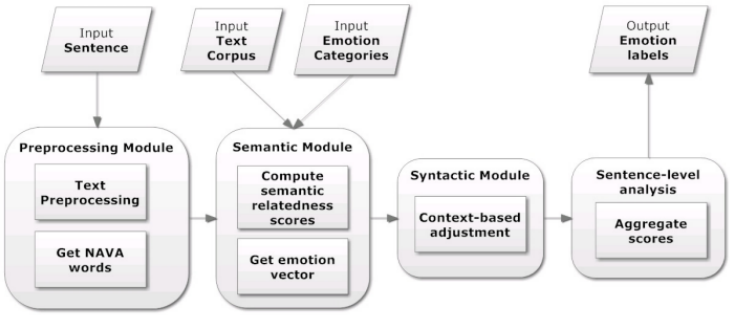

Figure 6: Overview of the emotion detection framework.

In this approach,[5] emotion detection is modelled as a classification problem where one or more nominal labels are assigned to a sentence from a pool of target emotion labels. The four main components of this framework include pre-processing, semantic, syntactic, and sentence analysis. The approach starts by extracting the affect words from the sentence, also called as NAVA (Noun Adjective Verb Adverb) words. Also, it is essential to look up the surrounding words in a sentence to get some idea about the emotion expressions of a particular word, for this, a syntactic dependency is employed. Focus is being made on three types of dependencies namely, adjectival complement, adjectival modifier, and negation modifier.

Usually, the emotion of a NAVA word can be defined as a vector whose elements each represent the strength of the affinity of the word for an emotion category. But if the sentence is deprived of an emotion keyword, it's difficult for a system to identify the emotion, so sematic relatedness is used to compute the emotion vector. It is observed that a meaning of a word can be found by observing it's usage across a large sample of language. If 2 words co-occur more often then they tend to be semantically related.[5] To calculate this, PMI is used. PMI between two words x and y is calculated as:

$$\text{PMI}(x,y) = \frac{\text{co-occurrence}(x,y)}{\text{occurrence}(x)\,\text{occurrence}(y)} \qquad \text{PMI}(w_i, e_j) = \sqrt[r]{\prod_{g=1}^{r} \text{PMI}\left(w_i, K_j^g\right)}$$

For computing emotion vector without context, PMI score between a word and the word representing an emotion concept has been used. Then the emotion vector of the dependent word is fine-tuned using context information. Lastly, the emotion vector of a sentence is calculated by aggregating the emotion vectors of all the affect words, and if the emotion vector score is above a threshold, it is classified as that emotion or else neutral.

To evaluate this framework, it was tested against 3 different corpus namely Wikipedia data, Gutenberg corpus, and Wiki-Guten. The authors also found

that [5] stemming does improve the accuracy of the emotion detection process and they used stemmed text for further experiments. This approach performs better than the keyword baseline and Alm's lextag methods and the context-based approach is slightly better than the context-free one. This method also proved significantly better than other methods on the ISEAR datasets with Four Emotions classification and seven emotions classification.

Algorithm	Joy	Sad	Ang-Dis	Fear	Avg.
Keyword baseline	0.371	0.270	0.346	0.328	0.328
LSA	0.103	0.106	0.631	0.071	0.227
PLSA	0.340	0.282	0.456	0	0.269
NMF	0.010	0.017	0.579	0.056	0.165
DIM	0.515	0.337	0.286	0.351	0.372
Context-based Wiki	0.564	0.408	0.628	0.592	0.548
Context-based Guten	0.574	0.253	0.582	0.536	0.486
Context-based W-G	0.542	0.296	0.668	0.574	0.520

Table 2: F-Score Results ISEAR Data Set Four Emotions.

Overall, the proposed method does not require annotated data or a detailed affect lexicon. The results show that the approach outperforms other recent unsupervised methods and is comparable to some supervised methods. However, the weakness of the approach is that the semantic relatedness scores depend on the text corpus from which they are derived. The study also found that the context-based approach consistently outperforms the context-free approach. In future research, the authors plan to derive semantic relatedness scores from multiple measures and test other syntactic dependencies.

```
# tokenize the text and get the POS tags for each word
tokens = nltk.word_tokenize(text)
pos_tags = nltk.pos_tag(tokens)

# define the NAVA categories
nava_categories = ['NN', 'NNS', 'NNP', 'NNPS', 'JJ', 'JJR', 'JJS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ', 'RB', 'RBR', 'RBS']

# filter the words by NAVA categories
affect_bearing_words = [word for word, tag in pos_tags if tag in nava_categories]

# lemmatize the affect-bearing words
lemmatizer = nltk.WordNetLemmatizer()
affect_bearing_words = [lemmatizer.lemmatize(word) for word in affect_bearing_words]

# remove stopwords and punctuation
stopwords = set(nltk.corpus.stopwords.words('english'))
affect_bearing_words = [word for word in affect_bearing_words if word not in stopwords and word.isalpha()]

# filter out words that are not found in WordNet
affect_bearing_words = [word for word in affect_bearing_words if len(wordnet.synsets(word)) > 0]

# print the affect-bearing words
print(affect_bearing_words)
```

```
['feel', 'happy', 'excited', 'today']
```

```
def get_emotion_vector(word):
    synsets = wn.synsets(word)
    if not synsets:
        return None
    synset = synsets[0]
    emotion_vector = [0] * len(emotions)
    scores = sia.polarity_scores(word)
    for i, emotion in enumerate(emotions):
        if scores['pos'] > scores['neg'] and emotion in ['joy', 'trust', 'anticipation']:
            emotion_vector[i] = 1
        elif scores['neg'] > scores['pos'] and emotion in ['sadness', 'fear', 'anger', 'disgust']:
            emotion_vector[i] = 1
    return emotion_vector
```

```
for word in affect_bearing_words:
    emotion_vector = get_emotion_vector(word)
    if emotion_vector is not None:
        print(f"{word.capitalize()} vector: {emotion_vector}")
```

```
Feel vector: [0, 0, 0, 0, 0, 0, 0, 0]
Happy vector: [0, 1, 0, 0, 1, 0, 0, 1]
Excited vector: [0, 1, 0, 0, 1, 0, 0, 1]
Today vector: [0, 0, 0, 0, 0, 0, 0, 0]
```

Code 2: Code Outputs.

3.2 Emotion Recognition from Text Based on Automatically Generated Rules

In this work, the authors propose a framework for emotion classification in English sentences where emotions are treated as generalized concepts extracted from the sentences. By looking at the syntactic and semantic structure of a sentence, an intermediate emotional data representation is generated which is later generalized to obtain emotion recognition rules (ERR). Then using classification algorithms, these generated ERRs are compared with the training set ERRs to get the emotion labels for the sentences. This paper aims to recognize six emotions as suggested by Ekman namely happiness, sadness, anger, fear, disgust, and surprise. Finally, to test out this approach, it was tested against the Twitter dataset and Aman dataset. In both cases, this approach proved to be better than the existing methods.

In this approach, emotions are treated as concepts extracted from the sentence and the representation of concepts depends on 2 principles; rules of composition are formed after removing the lexical parts of the sentence, and syntactic and semantic analysis must be performed to understand a sentence. The syntactic and semantic analysis is done by constructing the dependency parsing tree of the sentence and then pruning it as per some set rules. The output of this is basically the EER.

The authors describe 2 main phases of their methodology:[4] the offline phase, and the comparison and classification phase. The main role of the offline phase is to process each annotated sentence in the dataset by the Sentence Processor module to generate the corresponding EER.

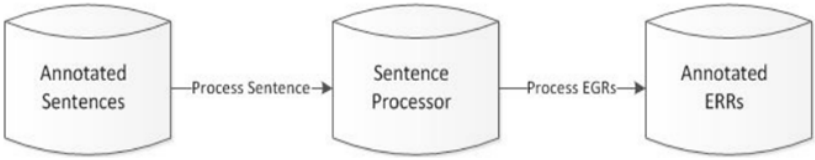

Figure 7: Offline Phase.

The sentence processor determines the POS tag for each word and after tagging, the Stanford dependency parser is applied which gives us the dependency tree.[4] Pruning of this tree is based on 2 rules: Separation Rules (2 rules), and Deletion Rules (3 rules). After pruning the tree, what we are left with is the required EER of the sentence.

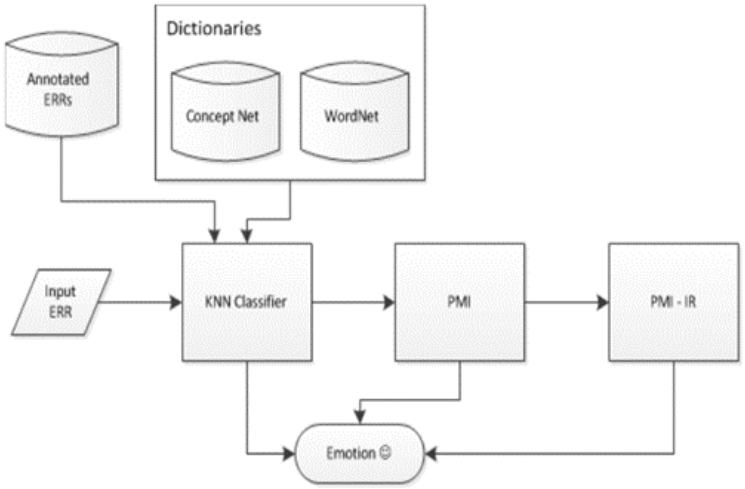

Figure 8: Classifier Overview

The next phase includes comparison and classification. The goal is to compare the two EERs i.e the input sentence EER and the EER generated from the offline phase. For comparison, a customized KNN algorithm is used. The KNN is customized in the sense that it is based on semantic and keyword similarity. However, if the input is rejected by the KNN classifier, an attempt is made to classify it using the PMI classifier, if it gets rejected by PMI as well, it is classified using PMI-IR. In the end, we will get a similarity score and the EER to which the input EER has the most similarity, the input is classified as per the emotion of the matched EER.

[4] This system follows a 2-tier client-server architecture where the client is responsible for getting input from the user and the server is responsible for processing it and sending back the results to the client. The method is tested on two different datasets. Firstly, on the Aman dataset, the training sentences were of length in the range of 0 to 10. The computation was compared against

the baseline approach which classifies based on keywords. It was found that [4] the average F-score of 84% was obtained on all the 6 emotions. For the second experiment consisting of the Twitter dataset, an F-score of 84% was obtained as well for the 5 emotions excluding disgust. A comparison was also made with the Emoheart classifier in both experiments.

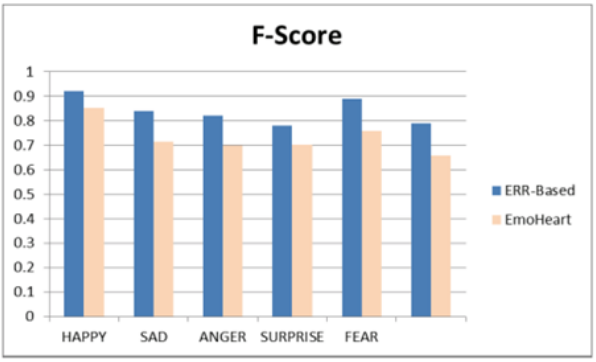

Figure 9: Comparison to EmoHeart on Aman dataset on all six emotions

Overall, the work presented a new approach to classifying emotions from textual data by performing complex syntactic and semantic analysis of the sentence and using ontologies such as Wordnet and ConceptNet. The approach was evaluated on two different datasets and outperformed the state-of-the-art method in emotion classification. The classifier was shown to be context-sensitive and able to generalize the training set for better coverage of emotion rules. The approach also demonstrated that comparing the relations between the words of the sentence could lead to better accuracy than assigning individual emotional rates for each word. The proposed classifier was shown to be flexible and easily extendable to classify any number of emotions by providing a reasonably-sized training set that covers the required emotions.

3.3 Emotion Detection in suicide notes

Suicide can be defined as death caused by self-directed injurious behavior with any intent to die, is a major public health concern and a leading cause of death worldwide. [6] Suicide prevention efforts focus on early risk recognition and referral support. However, due to information overload on social media, it's difficult to continuously monitor them. There is a need for automatic suicidal message detection. Most of the previous works on this is based on manual analysis and detection of important features that can distinguish between a fake note and a genuine one. Also, it is worth noting that suicide-related notes need not contain words related to suicide. The sentiment of such notes can also reside in a combination of words that might not have any suicide-related keywords.

This paper aims at the fine-grained detection of a large set of emotions (15) in suicide notes, where the emotions are detected at the sentence level. In order to do so, a supervised machine learning approach is used and trained an SVM on a manually labelled suicide note corpus.

[6] The dataset used consists of suicide notes written by 1319 people, collected between 1950 and 2011. Each note was annotated manually by at least 3 annotators. The most frequent labels found were, in order, instructions, hopelessness, love, information, and guilt. The data showed that most emotions were strongly lexicalized i.e. some words are typically associated with them.

The authors adopted a supervised machine learning methodology to predict the presence of one or more emotions in unseen data.[6] The data was preprocessed with MBSP with well defined features as follows: Lemmas, Lemmas + POS, Pruned lemmas + POS tags, Trigrams, WordNet synsets, SentiWordNet information, Subjective clues. To determine the optimal feature combinations, the 7 feature groups were combined into 17 feature sets. The dataset also contained spelling errors which could negatively affect the performance along the way. So, spelling were corrected and experiments were performed on both original and spellchecked data.

The 17 experimental feature sets, named A–Q on the horizontal axis. The presence of a feature in a feature set is indicated with an x (e.g. feature set F contains lemma and trigram bag-of-words features).

Feature set	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
Lemmas	x					x	x	x	x		x	x	x	x	x	x	x
Lemmas + POS		x															
Pruned lemmas + POS			x														
Trigrams				x		x				x	x				x	x	x
WordNet synsets					x		x					x	x		x		x
SentiWordNet								x				x		x		x	x
Subjectivity clues									x	x	x		x	x	x	x	x

Table 3: Hyperparameters chart

For the classification part, 15 binary classifiers were used whose outputs were combined. Since SVM is shown to work well in other NLP tasks, it was used. The results of the classifier are reported as F1-scores that place equal emphasis on both precision and recall. But in our case of finding suicide-related content, some precision can be traded up in favor of recall because a false negative is worse than a false positive. For this, bootstrap resampling has been used. It was also used to determine which threshold maximized F-score for each classifier.

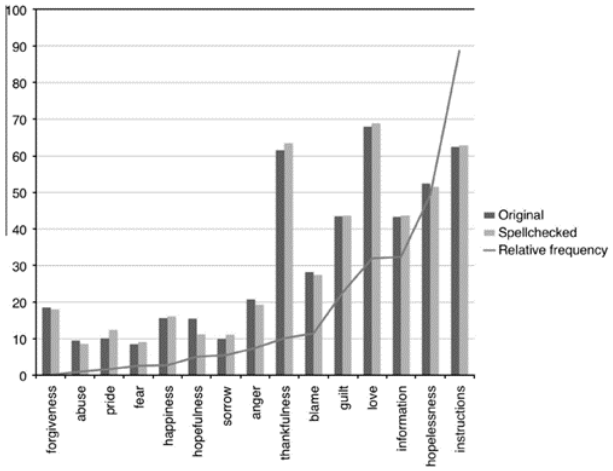

Figure 10: F-scores and relative frequency for each emotion

The bootstrap resampling experiments were conducted using SVM classifiers with 17 features set on 15 emotions.[6] For emotions with a frequency of more than 40 annotations per 1000 sentences, F-score was above 40% but it was worse for rare emotions as there were fewer training data for them. Emotions such as forgiveness, thankfulness, and love performed better than would be expected from frequency alone as they were lexicalized more often. Also, it was found that making spelling corrections was marginally beneficial. The 15 best classifiers use only 6 feature sets: trigrams and subjectivity clues (set J, 6classifiers), lemmas and trigrams (set F, 3 classifiers), lemmas, trigrams, and subjectivity clues (set K, 3 classifiers), lemma-POS pairs (set B, 1 classifier), WordNet (set E, 1 classifier) and lemmas, trigrams, WordNet, and subjectivity clues (set O, 1 classifier). The final system output is produced by aggregating the outputs of the best performing classifier for each emotion.

Overall, the paper outlines a machine-learning approach for detecting emotions in suicide notes, using support vector machines and a variety of lexical and semantic features.[6] Results suggest that the most successful features are trigrams and lemma bags-of-words and subjectivity clues, but data sparseness in rare emotions remains a challenge. The paper suggests that deeper semantic analysis and more advanced spelling correction could yield informative features for emotion classification. The detection of emotions in suicide notes has applications in forensic linguistics and suicide prevention, and the paper suggests that future research could use the presence of emotions as features for predicting suicidality in text.

4 Summary of papers between 2015 & 2019

4.1 Semantic Emotion Neural Network for Emotion Recognition from Text

The paper proposes a novel neural network architecture called the Semantic-Emotion Neural Network (SENN) for emotion recognition in text. The goal of the model is to utilize both semantic/syntactic and emotional information to better understand the emotional content of text. SENN consists of two sub-networks: a bidirectional Long-Short Term Memory (BiLSTM) network for semantic encoding and a convolutional neural network (CNN) for emotion encoding. The two sub-networks are fed the same sequence of words and generate two matrices of word embeddings: $Z_sem for semantic embedding and Z_emo for emotion embedding.$

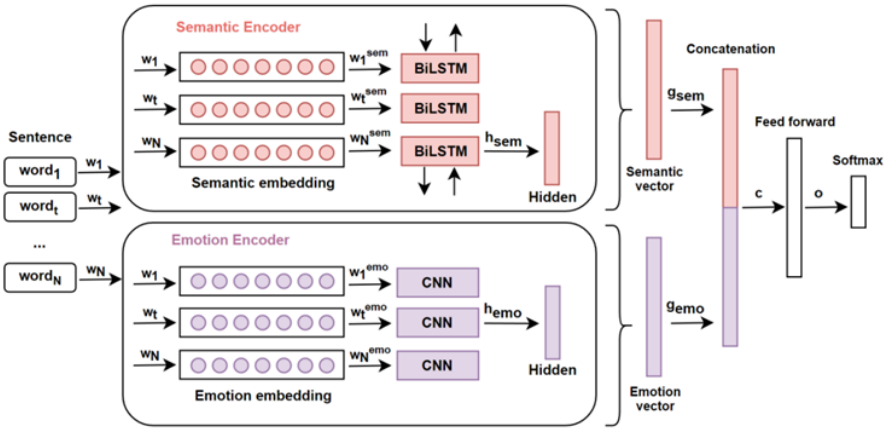

Figure 11: SENN model architecture.

$$Z_{emo} = [w_1^{emo}, \ldots, w_t^{emo}, \ldots, w_N^{emo}] \in R^{N \times d}$$
$$Z_{sem} = [w_1^{sem}, \ldots, w_t^{sem}, \ldots, w_N^{sem}] \in R^{N \times d}$$

$$w_t^{emo} = [e_{t1}, \ldots, e_{tk}, \ldots, e_{td}]$$
$$w_t^{sem} = [s_{t1}, \ldots, s_{tk}, \ldots, s_{td}]$$

Figure 12: Word matrices & word vectors.

To better extract emotion features from emotion-based word embeddings, the CNN is used to apply convolving filters to local features.[10] The word embedding vectors are concatenated as the feature vector v of the sequence. The final emotion encoding and semantic encoding are concatenated and fed into a feedforward layer with a softmax classifier to predict the emotion of the text.

Parameter	CNN	BiLSTM
Learning rate	0.001	0.001
Batch size	128	128
Hidden dimension	256	-
Number of layers	2	-
Number of filters	-	100
Filter sizes	-	[3, 4, 5]
Early stopping patience	20	20
Dropout	0.5	0.5

Table 4: Hyperparameter setting.

The SENN model was evaluated on ten emotion-annotated datasets from multiple domains, including dialogues, tweets, fairy tales, blogs, and news headlines. The model was compared against various baseline models, including Naïve Bayes, Random Forest, Support Vector Machine, Logistic Regression, Convolutional Neural Network, Long-Short Term Memory (LSTM), GRU, RCNN, and CNN+LSTM. The evaluation was based on precision, recall, and F1-score.

Model		D	C	T	TE	I	E	ET	G	EC	S
NB	BOW	73.2	44.7	49.0	48.2	68.7	83.7	49.9	55.7	71.0	58.4
	TFIDF	61.7	40.0	34.9	24.6	67.8	74.5	44.3	49.7	54.1	58.4
RF	BOW	77.8	43.8	48.1	40.1	63.7	86.9	47.6	54.6	94.6	58.4
	TFIDF	77.0	43.8	46.5	38.2	64.9	86.4	51.0	52.6	92.0	58.0
SVM	BOW	80.1	47.5	53.5	52.6	63.2	86.2	52.4	48.8	95.1	58.4
	TFIDF	78.5	46.1	55.0	54.1	65.9	88.1	52.3	50.6	95.5	57.0
LR	BOW	79.8	46.1	53.8	49.0	69.9	88.2	52.8	53.0	96.9	54.3
	TFIDF	80.0	46.8	55.6	49.8	69.5	89.2	51.6	50.0	**98.7**	53.3
CNN	Word2Vec	81.8	50.6	59.1	57.7	73.1	89.0	51.2	50.7	97.8	54.3
	GloVe	81.9	49.1	57.6	57.3	73.5	89.1	52.1	57.7	96.3	53.9
	FastText	82.6	50.2	58.5	**62.3***	73.9	89.7	52.5	48.3	95.9	55.5
	EWE	81.7	50.6	59.3	58.4	73.6	86.4	48.4	56.0	95.9	56.5
GRU	Word2Vec	80.5	45.7	54.2	46.0	67.8	87.9	53.7	52.9	96.4	58.7
	GloVe	80.5	46.1	56.0	46.3	25.5	87.8	54.7	48.9	96.8	57.1
	FastText	81.9	47.5	53.5	33.3	70.3	88.0	51.8	54.2	95.9	63.5
	EWE	81.5	46.3	54.2	40.5	71.5	86.4	53.1	55.5	97.3	57.6
BiGRU	Word2Vec	81.0	48.4	55.8	56.8	71.6	89.3	49.8	50.6	95.9	57.7
	GloVe	83.3	48.6	57.6	53.9	71.8	89.1	47.7	54.4	96.4	61.8
	FastText	82.8	48.0	56.6	56.1	72.3	89.2	51.7	56.1	96.9	57.0
	EWE	83.2	49.8	55.8	57.0	71.1	89.7	46.8	54.3	95.9	**67.4**
LSTM	Word2Vec	78.5	45.8	51.9	43.2	23.7	87.0	50.1	50.4	93.7	58.7
	GloVe	79.8	45.3	54.3	34.2	69.3	88.2	51.7	53.0	96.9	58.8
	FastText	63.5	45.7	54.4	50.5	25.4	88.4	51.6	51.1	92.0	58.4
	EWE	77.0	42.9	55.3	35.3	21.8	88.8	51.1	51.7	94.0	55.9
BiLSTM	Word2Vec	80.2	49.3	54.6	53.1	71.0	89.7	51.0	53.5	93.1	55.3
	GloVe	82.9	48.6	57.4	53.5	72.2	89.8	51.3	51.6	97.3	65.0
	FastText	82.2	48.5	56.2	55.8	70.2	89.2	49.0	51.1	95.4	57.0
	EWE	83.7	48.4	58.4	55.8	72.0	88.1	51.6	52.9	95.5	49.2
RCNN	Word2Vec	80.8	50.3	57.0	53.2	72.1	89.3	52.5	51.6	98.2	61.8
	GloVe	83.2	49.5	56.9	61.0	70.8	88.5	53.3	52.8	**98.7**	60.3
	FastText	81.8	49.6	57.4	59.2	73.3	89.6	52.4	51.8	97.8	55.9
	EWE	83.6	48.0	57.3	56.2	73.1	88.1	**55.4**	50.5	95.5	58.0
CNN LSTM	Word2Vec	82.1	49.7	56.4	56.9	71.6	87.6	47.6	49.6	96.4	55.4
	GloVe	83.3	49.8	56.5	55.4	72.4	88.6	49.4	56.0	97.3	55.3
	FastText	82.2	**51.0**	57.9	54.9	72.2	89.9	48.5	48.7	95.9	46.7
	EWE	82.7	50.0	57.3	54.9	73.5	89.0	49.3	52.3	95.9	48.1
SENN	Word2Vec + EWE	84.2	50.8	**59.7**	60.3	73.7	**91.0***	**55.4**	**58.4**	96.9	65.0
	GloVe + EWE	**84.3**	50.5	**59.7**	60.9	**74.6***	**90.4**	54.6	**59.3***	**98.8***	64.2
	FastText + EWE	**84.8***	**51.1***	**61.3***	**61.7**	**74.5**	**90.4**	**56.3***	55.6	98.3	**70.8***

Table 5: Comparison with the baseline models

The experimental results show that the proposed model achieves superior performance compared to other state-of-the-art approaches for emotion recognition in text.[10] SENN outperforms the other models on nine out of the ten datasets, with F1-scores ranging from 51.1% to 84.8%. The proposed model can also be further improved by using other emotional word embeddings. Overall, the paper presents a promising approach for emotion recognition in text that integrates both semantic and emotional information.

```
In [24]: # Evaluate the model
         score = model.evaluate(X_test, y_test, verbose=0)
         print('Test loss:', score[0])
         print('Test accuracy:', score[1])

         Test loss: 0.9536104202270508
         Test accuracy: 0.8233638405799866
```

Code 3: Code Output

4.2 Knowledge-Enriched Transformer for Emotion Detection in Textual Conversations

The paper proposes a Knowledge-Enriched Transformer (KET) model to detect emotions in textual conversations. The main challenge in this task is the reliance on context and common sense knowledge to express emotions. KET addresses this challenge by leveraging hierarchical self-attention and dynamic context-aware affective graph attention mechanisms to model the structure of conversations and refer to external knowledge bases.

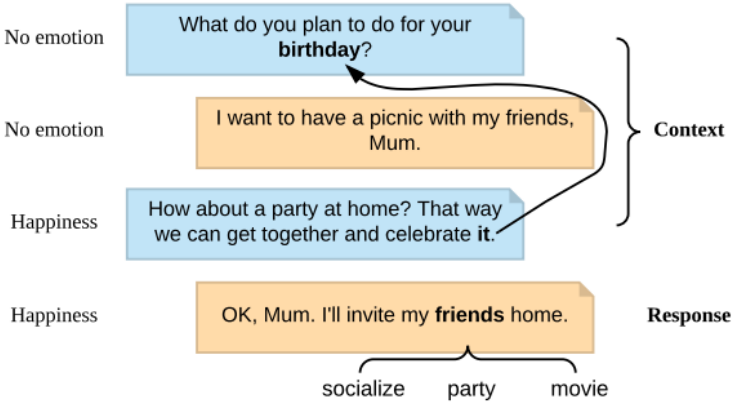

Figure 13: Example Conversation

The authors start by explaining the Transformer, a powerful NLP model that captures intra-sentence and inter-sentence correlations using self-attention and cross-attention modules.[12] The shorter path of information flow in the Transformer compared to gated RNNs and CNNs allows KET to model contextual information more efficiently.

[12] The main task of the KET model is to maximize the following function:

$$\Phi = \prod_{i=1}^{N} \prod_{j=1}^{N_i} p(Y_j^i | X_j^i, X_{j-1}^i, ..., X_1^i; \theta),$$

The paper then introduces the hierarchical self-attention mechanism, which allows KET to model the hierarchical structure of conversations. The authors propose two steps: first, an utterance-level self-attention layer computes the representation of each utterance, and second, a context-level self-attention layer computes the representation of the entire conversation from the learned utterance representations.

To exploit commonsense knowledge, KET leverages external knowledge bases such as ConceptNet and NRC VAD. ConceptNet is a knowledge base with nodes representing concepts and edges representing relations between them. NRC VAD is an emotion lexicon with valence, arousal, and dominance scores for around 20k English words. KET retrieves a connected knowledge graph for each non-stopword token and computes a dynamic context-aware affective graph attention mechanism to enrich word embedding with concept representations.

Dataset	Domain	#Conv. (Train/Val/Test)	#Utter. (Train/Val/Test)	#Classes	Evaluation
EC	Tweet	30160/2755/5509	90480/8265/16527	4	Micro-F1
DailyDialog	Daily Communication	11118/1000/1000	87170/8069/7740	7	Micro-F1
MELD	TV Show Scripts	1038/114/280	9989/1109/2610	7	Weighted-F1
EmoryNLP	TV Show Scripts	659/89/79	7551/954/984	7	Weighted-F1
IEMOCAP	Emotional Dialogues	100/20/31	4810/1000/1523	6	Weighted-F1

Table 6: Datasets Information

The paper evaluates KET on five emotion detection datasets of various sizes and domains: EC, DailyDialog, MELD, EmoryNLP, and IEMOCAP. The authors compare KET with several baselines, including cLSTM, CNN, CMM+cLSTM, BERT BASE, DialogueRNN, KET SingleSelfAttn, and KET StdAttn.

Dataset	Domain	#Conv. (Train/Val/Test)	#Utter. (Train/Val/Test)	#Classes	Evaluation
EC	Tweet	30160/2755/5509	90480/8265/16527	4	Micro-F1
DailyDialog	Daily Communication	11118/1000/1000	87170/8069/7740	7	Micro-F1
MELD	TV Show Scripts	1038/114/280	9989/1109/2610	7	Weighted-F1
EmoryNLP	TV Show Scripts	659/89/79	7551/954/984	7	Weighted-F1
IEMOCAP	Emotional Dialogues	100/20/31	4810/1000/1523	6	Weighted-F1

Table 7: Comparative Analysis

[12] The results show that KET outperforms the baselines on most datasets, demonstrating its robustness across different training sizes, context lengths, and domains. KET SingleSelfAttn and KET StdAttn perform comparably with the best baselines on all datasets except IEMOCAP. However, both variants perform noticeably worse than KET on all datasets except EC, validating the importance of the proposed hierarchical self-attention and dynamic context-aware affective graph attention mechanisms.

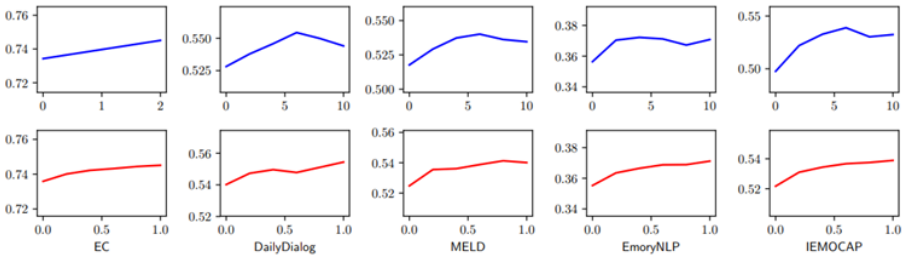

Figure 14: Validation performance by KET

In conclusion, the paper presents a knowledge-enriched transformer model for emotion detection in textual conversations. KET leverages hierarchical self-attention and dynamic context-aware affective graph attention mechanisms to model the structure of conversations and external knowledge bases to enrich word embedding with concept representations. The results show that both contextual information and common sense knowledge are beneficial to the model's performance.

4.3 Emotion Detection From Text using PMI Word based approach

This paper explores the role of emotion detection in the field of artificial intelligence, particularly in developing human-machine interfaces. It discusses various techniques used to detect emotions in humans, including facial expressions, body movements, blood pressure, heart rate, and textual information. The paper focuses specifically on emotion detection from textual information since there is an immense amount of textual data available on the internet.[11] Extracting emotion from textual data can be useful in various fields, such as emotion marketing, where emotions like brand, individuality, and prestige are essential for purchasing decisions.

The paper examines four primary methods of detecting emotions from text, namely keyword-based detection, learning-based detection, lexical affinity method, and hybrid detection. Hybrid detection, which combines two or more methods, is the most likely method to produce accurate results. However, the main challenge in this method is to find the most effective combination of methods.

The paper also discusses the challenges involved in emotion detection, such as data collection, feature choices, labeling of emotions, and machine learning classifiers. To address these challenges, the paper proposes a method that combines both keyword-based and learning-based methods to achieve high accuracy in detecting emotion from text.

The proposed method consists of two approaches: a word-based approach and a learning-based approach.[11] The word-based approach uses the Natural Language Toolkit (nltk) package, which is best for analyzing human language data. The approach involves assigning emotion labels to detect emotions like joy, fear, anger, sadness, and happiness from the text. Negation words are also assigned to detect negative emotions. The approach involves removing unnecessary characters from the sentences, tagging the words to segregate them into different categories, stemming the tagged words to create a data frame, and creating a WordNet between the words in column 2 and 3. The new lines whose emotions are to be detected are then passed, and the result is obtained.

The learning-based approach involves using the Twitter Graph API to extract tweets, saving them in an excel file with two columns (author and tweet), and creating a dataset for training and testing the machine. The dataset contains a column that predefines the emotions. The training and testing datasets are divided into a 3:1 ratio, and the machine is trained and tested using the training dataset.

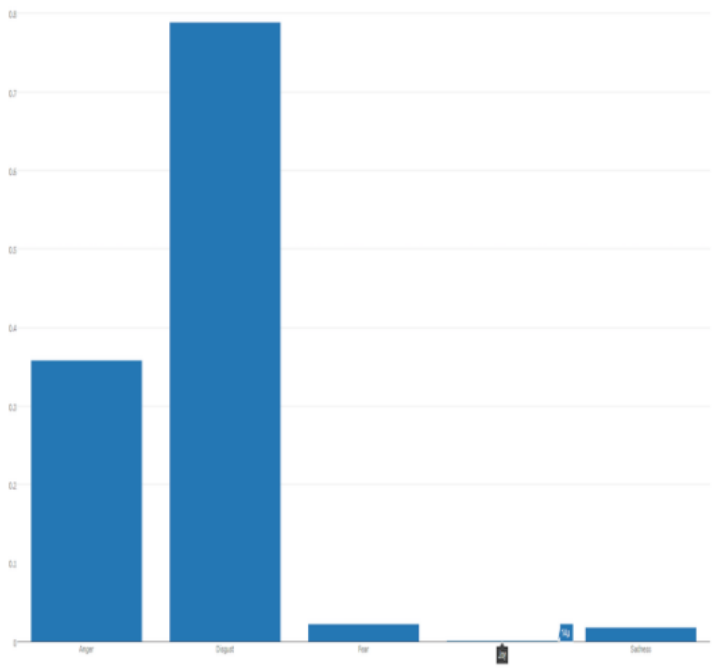

Figure 15: Graphical representation of Emotion from text

The paper concludes by discussing the results and discussions of the proposed method. The proposed method detects emotion from the text by taking input through voice and converting it into text using the Speech Recognition Package in Python. The program then cleans the data, removes all blank spaces and meaningless words, and tokenizes the words to find the emotion

from the sentence. The output is shown in graphical and coordinate forms.

Overall, the paper provides an overview of the importance of emotion detection in artificial intelligence, particularly in human-machine interfaces. It also discusses the challenges involved in emotion detection and proposes a method that combines both keyword-based and learning-based methods to achieve high accuracy in detecting emotion from text. The proposed method has the potential to be useful in various fields, including emotion marketing.

5 Summary of Papers between 2020 & 2023

5.1 Deep learning based Text Emotion Recognition for Chatbot applications

The paper focuses on the recognition of emotions in textual conversations, which is important as many human interactions take place through text devices. The study uses a deep learning-based Long Short-Term Memory (LSTM) mechanism for the identification of emotions in textual conversations. The dataset used in the study is the 'Emotion Classification' dataset, which includes six emotional groups. The study concludes that the LSTM-based text emotion classification provides relatively higher accuracy compared to existing learning methods.

The research focuses on casual interactions with users in the context of an online chatbot. The proposed method involves pre-treating the dataset by omitting punctuations, transforming words to lowercase, and eliminating punctuation marks. Every word in the expression is represented by a unique integer. The preprocessing phase also calculates the actual word count in the training data and the maximum word count in a single sentence. To do the classification, SVM's benchmark method needs feature extraction, and the study uses TF-IDF as features for the SVM.

The paper proposes a hybrid deep learning model to automatically learn features and recognize all existing emotions in the text, and to determine the predominant emotion of the sentence only with minimal feature technology. The proposed multilabelling approach involves segmentation of sentences, word representation using word embedding methods, and primary determination of emotion by the sentence using an LSTM network.

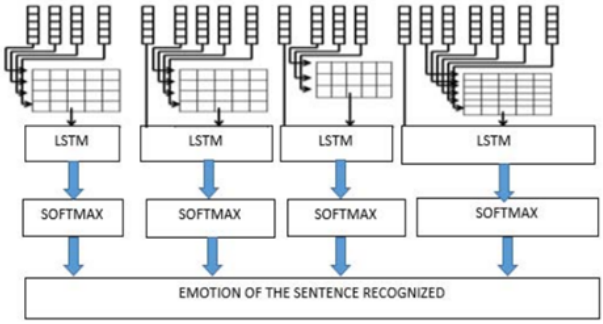

Figure 16: Deep Model Structure

In the fig , [8] a LSTM simulation diagram using KERAS Software is presented. The following layers form LSTM for emotion classification: Embedding layer, LSTM stratum, and output layer. There is only one neuron to the embedding layer. The word transmitted into this neuron will be converted into a true valued vector (output dimension) of required lengths. Once the network has been equipped we will obtain the masses of the embed layer.

$$c_{j,i} = f(x_{i:i+j-1}.K_j^T + b) \quad C_j = \frac{1}{m} \sum_{i=1}^{m} c_{j,i}$$

Figure 17: Convolution formula

The study transforms words into continuous vector representations using word embedding methods, and each component can be interpreted as a matrix of embedding. The study uses three convolutionary filters to create dense representations of each component. The first, second, and third filters represent unigrams, bigrams, and trigrams, respectively.[8] Convolution performs on the embedding matrix, and the average pooling layer combines the varying number of features from the convolution layer into a vector. The concatenation of three filters is used as part representation and decoded into probabilities by a softmax layer for each group of emotions. In the last step, LSTM is used to assess the prevailing emotion of a sentence.

Group	Efficiency	Sensitivity	F1-Ranking
Rage	95.9%	94.8%	95.1%
Worry	95.48%	95.33%	95%
Happiness	90.4%	91.24%	90.13%
Affection	96.36%	95.62%	95.4%
Sadness	96.5%	95.78%	96.50%
Astonishment	94%	93.25%	94%
Kindness	94.8%	93.41%	93%
Average	**94.7%**	**94.2%**	**94.1%**

Table 8: Efficiency, Recall, and F1-Score of LSTM

Group	Efficiency	Sensitivity	F1-Ranking
Rage	97.5%	96.4%	97.5%
Worry	95.2%	94%	94.9%
Happiness	95.63%	95.46%	95%
Affection	93.1%	92.6%	93.35%
Sadness	90.7%	91.6%	90%
Astonishment	91.3%	90%	90.13%
Kindness	90.4%	91.16%	90.1%
Average	**93%**	**93.03%**	**92.9%**

Table 9: Efficiency, Recall, and F1-Score of SVM

The testing results show that the LSTM achieves an overall accuracy of 94.15%, while the Nested LSTM model achieves an overall accuracy of 92.34%. The SVM method's average performance is 93.1%. The study concludes that LSTM, SVM, and Nested LSTM methods can be used to identify emotions in multiclass based on the results of the discussion and evaluation conducted in the previous section.[8] LSTM has the best accuracy among accuracy methods and the best average performance in terms of efficiency, sensitivity, and f1score at 94.7%, 94.2%, and 94.1% respectively.

In summary, the paper highlights the importance of recognizing emotions in textual conversations and proposes a deep learning-based LSTM mechanism for the identification of emotions. The study uses the 'Emotion Classification' dataset and shows that the proposed LSTM-based method provides relatively higher accuracy compared to existing learning methods. The study also proposes a hybrid deep learning model that automatically learns features and recognizes all existing emotions in the text, and determines the predominant emotion of the sentence. The testing results show that LSTM has the best accuracy and the best average performance in terms of efficiency, sensitivity, and f1score.

5.2 Emotion Recognition from Text Stories Using an Emotion Embedding model

The paper focuses on emotion recognition from text stories using an emotion embedding model. Emotions are an essential component of stories, as they are closely associated with characters and the sequence of events that make up the plot. Therefore, emotion analysis is crucial for text story research, including text story generation, text story understanding, and box office predictions based on textual film information.

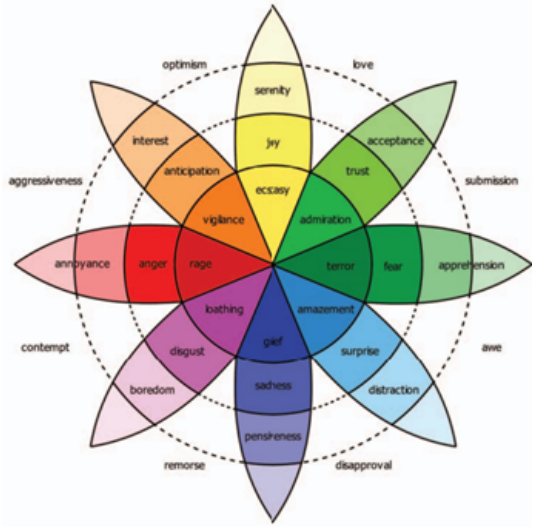

Figure 18: Plutchik's Wheel of Emotions Model

To build an emotion embedding model, the authors collected 144,701 tweets, each of which was given an emotional hashtag. They then used these emotion hashtags as emotion labels and built a convolutional neural network (CNN) model for emotion classification. During the learning process, they extracted the embedding model and word embedding layer created during the emotion classification learning process. They defined this as an 'emotion embedding model' and applied it to classify story text emotions.

The authors used the [9] ROC Stories dataset, which includes 52,666 stories, each consisting of five simple sentences (a total of 263,330 sentences). They used the NLTK VADER Sentiment Analyzer to detect emotional words in the story sentences.

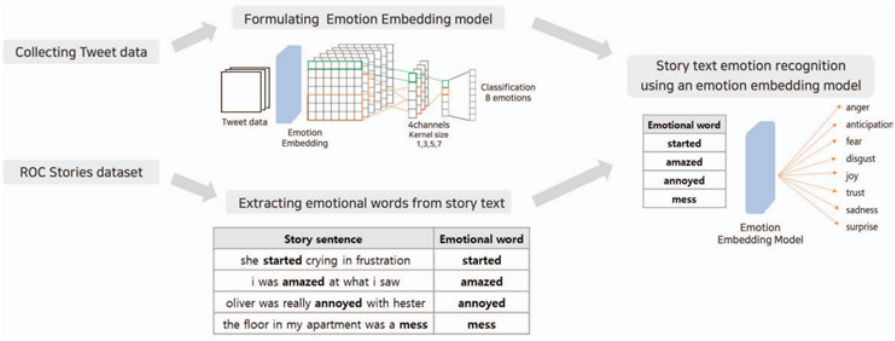

Figure 19: Emotion Embedding Approach to Textual Story Emotion Classification

To extract emotions from text stories using the emotion embedding model, the authors computed the cosine similarity between the selected emotional words and emotional hashtags for emotion annotation in tweet data. They [9] analyzed a total of 137,052 story text sentences and found that Joy, Sadness, Fear, and Anger were the top four emotions, occupying a majority of 73.55% of the total counts. These four emotions showed a higher proportion because the tweet data used in creating the embedding model also had a lot of Joy, Sadness, Fear, and Anger data (74.26%).

Result		
Emotion	Count	Percent
Anger	19711	14.4
Anticipation	13024	9.5
Disgust	3251	2.37
Fear	22480	16.4
Joy	31371	22.89
Trust	6769	4.94
Sadness	27245	19.88
Surprise	13201	9.63

Table 10: Emotion Analysis Result Of Story Text

Emotion	Accuracy	Fleiss' kappa
Anger	0.367	0.216
Anticipation	0.567	0.363
Disgust	0.55	0.342
Fear	0.483	0.282
Joy	0.733	0.551
Trust	0.517	0.352
Sadness	0.45	0.281
Surprise	0.433	0.293

Table 11: Human Analysis Results - Accuracy And Fleiss' Kappa Score

To evaluate the emotional embedding model's performance, the authors randomly selected 120 sentences (15 sentences for each of the eight emotions) from the story sentences. Four human raters evaluated the emotions of the sample sentences. [9] Based on the evaluation of human raters, the Joy emotion showed the highest accuracy (=.733), while Anger resulted in the lowest accuracy value (=.367) and the lowest Kappa score (=.216). Inspection of some Anger labeled sentences suggests that these sentences often accompanied other negative emotions such as Sadness and Fear. On the other hand, sentences labeled as Joy did not accompany any other positive emotion.

In conclusion, the authors present a method to extract the emotion of a sentence using an emotion embedding model. They built an emotion embedding word model using the collected tweet data annotated with hashtags and extracted the representative emotional word in each sentence of the ROC story data. The representative emotional word was then used to classify the

emotion of the sentence leveraging the cosine similarity. The authors conducted experiments, and the results show that their approach is promising.

One limitation of this approach is that it does not consider the contextual information of the story, as the authors analyzed emotions from story texts based on the emotional words representing each story sentence. Future research could explore incorporating contextual information into the emotion recognition process to enhance accuracy. Additionally, the authors note that the emotion embedding model could be further optimized by incorporating other types of emotional data, such as images or videos, into the training process.

5.3 Emotion Recognition by Textual Tweets Classification Using Voting Classifier (LR-SGD)

The research paper discusses the task of sentiment analysis on Twitter, which involves analyzing the attitude, emotions, and opinions of different people towards anything by employing machine learning models. Sentiment analysis is essential for understanding public opinion on news, policies, social movements, and personalities, which could be valuable for businesses and governments. The paper presents a comparative performance analysis of seven machine learning models for emotion recognition by classifying tweets as happy or unhappy. The models include support vector machine, decision tree classifier, Naive Bayes, random forest, gradient boosting machine, logistic regression, and a proposed voting classifier (LR-SGD) with TF-IDF.[7] The proposed model outperformed other models with 79% accuracy and 81% F1 score. To validate the stability of the proposed approach, it was applied to two more datasets, one binary and another multi-class dataset, and achieved robust results.

The Twitter dataset used in this study was obtained from Kaggle repository, which contained 99989 records labeled as happy and unhappy according to their sentimental polarity using symbol 1 and 0. The dataset was pre-processed by removing unnecessary data, including unstructured or semi-structured data, which can increase the training time of the model and degrade its performance. The pre-processing steps included tokenization, case-conversion, stopwords removal, and removal of numbers. After pre-processing, the textual features were converted into vector form using TF and TF-IDF techniques.

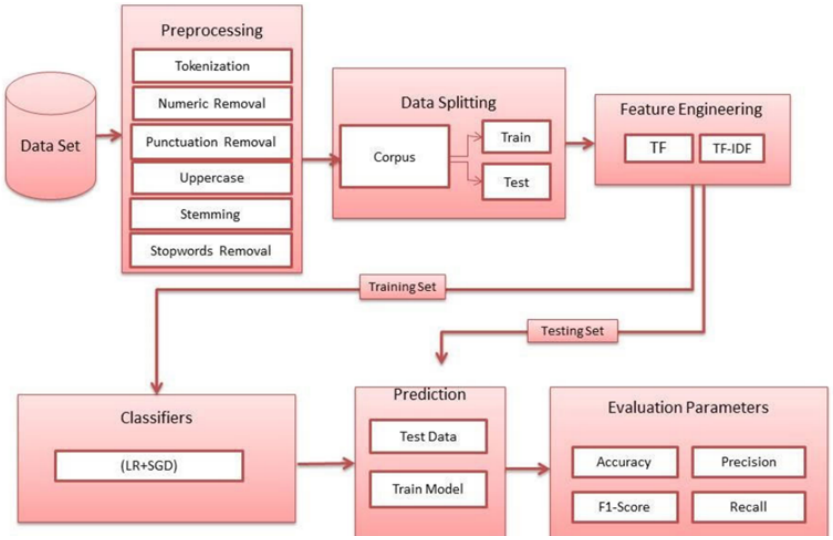

Figure 20: Proposed methodology architecture diagram.

[7] The proposed methodology of the research work utilized five supervised machine learning algorithms: SVM, Naive Bayes, Random Forest, Decision Tree, Gradient Boosting model, Logistic Regression, and a Voting Classifier (Logistic Regression + Stochastic Gradient Descent classifier). SVM is a popular algorithm that separates the classes by finding the hyperplane that maximizes the margin between them. Naive Bayes is a probabilistic classifier that assumes independence between features. Random Forest is an ensemble method that constructs a forest of decision trees to improve accuracy and reduce overfitting. Decision Tree is a classification algorithm that recursively splits the data into subsets based on the most significant feature. Gradient Boosting Machine is a boosting model that works by a model formed by an ensemble of weak prediction models, commonly decision trees. Logistic Regression estimates class probabilities based on output and predicts if the input is from class X or Y. Stochastic Gradient Descent is an iterative strategy for optimizing a target function with appropriate perfection properties. It calculates the degree of advancement based on the development of alternative variables.

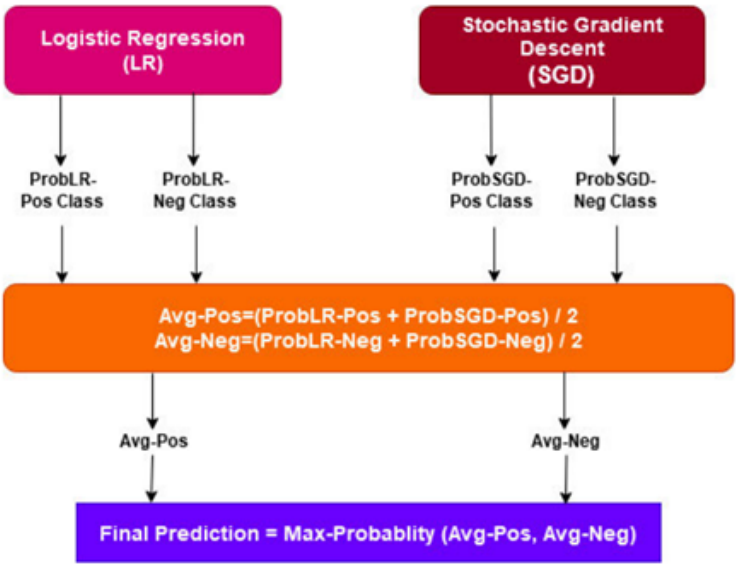

Figure 21: Proposed voting classifier architecture (LR-SGD)

[7] The Voting Classifier is a meta-classifier that combines multiple individual classifiers and their predictions to achieve better performance than a single classifier. The VC is a cooperative learning technique that uses majority voting to classify data. It has been shown that the combination of multiple classifiers could be more effective than any single one. The proposed model LR-SGD with TF-IDF achieved the best performance in terms of accuracy and F1 score, indicating its effectiveness in sentiment analysis on Twitter.

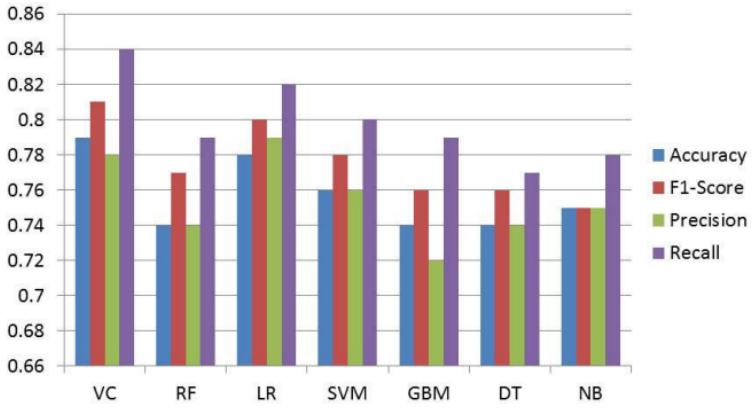

Figure 22: Classification result comparison of all machine learning models using TF features.

The research paper contributes to the field of sentiment analysis by providing a comparative analysis of different machine learning models for emotion recognition by tweet classification using TF and TF-IDF. It proposes a voting

classifier and validates its stability by applying it to two more datasets, one binary and another multi-class dataset. The proposed model could assist businesses and governments in understanding public opinion on products, policies, events, and personalities, which could be valuable for decision-making. The study highlights the importance of feature engineering in sentiment analysis and recommends using TF-IDF for better performance.

References

[1] Chung-Hsien Wu, Ze-Jing Chuang, and Yu-Chung Lin. 2006. Emotion recognition from text using semantic labels and separable mixture models. ACM Transactions on Asian Language Information Processing 5, 2 (June 2006), 165–183. https://doi.org/10.1145/1165255.1165259

[2] Chuang, Ze-Jing Wu, Chung-Hsien. (2002). Emotion recognition from textual input using an emotional semantic network. 10.21437/ICSLP.2002-558.

[3] Saima Aman and Stan Szpakowicz. 2008. Using Roget's Thesaurus for Fine-grained Emotion Recognition. In Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I.

[4] S. Shaheen, W. El-Hajj, H. Hajj and S. Elbassuoni, "Emotion Recognition from Text Based on Automatically Generated Rules," 2014 IEEE International Conference on Data Mining Workshop, Shenzhen, China, 2014, pp. 383-392, doi: 10.1109/ICDMW.2014.80.

[5] A. Agrawal and A. An, "Unsupervised Emotion Detection from Text Using Semantic and Syntactic Relations," 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Macau, China, 2012, pp. 346-353, doi: 10.1109/WI-IAT.2012.170.

[6] Desmet, B., Hoste, V. (2013). Emotion detection in suicide notes. EXPERT SYSTEMS WITH APPLICATIONS, 40(16), 6351–6358. https://doi.org/10.1016/j.eswa.2013.05.050

[7] A. Yousaf et al., "Emotion Recognition by Textual Tweets Classification Using Voting Classifier (LR-SGD)," in IEEE Access, vol. 9, pp. 6286-6295, 2021, doi: 10.1109/ACCESS.2020.3047831.

[8] M. Karna, D. S. Juliet and R. C. Joy, "Deep learning based Text Emotion Recognition for Chatbot applications," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 988-993, doi: 10.1109/ICOEI48184.2020.9142879.

[9] S. -H. Park, B. -C. Bae and Y. -G. Cheong, "Emotion Recognition from Text Stories Using an Emotion Embedding Model," 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), Busan, Korea (South), 2020, pp. 579-583, doi: 10.1109/BigComp48618.2020.00014.

[10] E. Batbaatar, M. Li and K. H. Ryu, "Semantic-Emotion Neural Network for Emotion Recognition From Text," in IEEE Access, vol. 7, pp. 111866-111878, 2019, doi: 10.1109/ACCESS.2019.2934529.

[11] Ramalingam, V. V., et al. "Emotion detection from text using PMI Word based approach." Journal of Physics: Conference Series. Vol. 1000. No. 1. IOP Publishing, 2018.

[12] Zhong, Peixiang, Di Wang, and Chunyan Miao. "Knowledge-enriched transformer for emotion detection in textual conversations." arXiv preprint arXiv:1909.10681 (2019).