A Project Report

NODE CLASSIFICATION USING GNN

MINOR PROJECT II

Pranjal Pateriya(9920103188)

Lavish Arora(9920103193)

Manas Tripathi(9920103196)

Under the supervision of:

Ms. Anuradha Gupta

Department of CSE/IT

Jaypee Institute of Information Technology University, Noida

April 2023

Department of CSE/IT JIIT, Noida

2023

CERTIFICATE

This is to certify that the work titled "Node Classification Using Graph Neural Network " submitted by Pranjal Pateriya:(9920103188), Lavish Arora:(9920103193) and Manas Tripathi(9920103196) of B.Tech of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of any other degree or diploma.

Project Mentor
(Ms. Anuradha Gupta)
Dept. of CSE/IT
JIIT , Noida

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and beliefs, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma from a university or other institute of higher learning, except where due acknowledgment has been made in the text.

Pranjal Pateriya (9920103188)
Lavish Arora (9917103193)
Manas Tripathi (9917103196)

Noida
25-April-2023

Abstract

Node classification is a fundamental problem in graph analytics, where the objective is to predict the class label of a node in a given graph. Graph Neural Networks (GNNs) have emerged as a powerful approach for node classification, which can leverage both the graph structure and node features to learn informative node embeddings. In this project, we propose to investigate the use of GNNs for node classification on real-world datasets. Specifically, we will focus on three key aspects of GNNs: word embeddings, graph partitioning, and Graph Convolutional Networks (GCNs). First, we will explore the use of word embeddings to represent the node features in a graph. Word embeddings have been widely used in natural language processing (NLP) to capture semantic and syntactic relationships between words. We will adapt this technique to learn embeddings for nodes in a graph, which can then be used as input features for GNN models. Second, we will investigate the impact of graph partitioning on the performance of GNN models. Graph partitioning is a technique for dividing a graph into multiple subgraphs, which can be processed in parallel to improve efficiency. We will explore various graph partitioning algorithms, such as METIS. Finally, we will focus on Graph Convolutional Networks (GCNs), a popular type of GNN that can capture local and global graph structures by performing graph convolutions on the node features.

Acknowledgement

We express our sincere gratitude to Ms. Anuradha Gupta, Department of CSE,Jaypee Institute of Information Technology, Noida for her invigorating guidance, continual encouragement, and supervision throughout the present work. We also wish to extend our thanks to our batch mates for their insightful comments and constructive suggestions to improve the quality of this project work..

Credentials of students:

Pranjal Pateriya

Lavish Arora

Manas Tripathi

Contents

ABBREVIATIONS

- **WORD2VEC:**Word to Vector
- **GNN:**Graph Neural Network
- **GCN:**Graph Convolutional Networks
- **BERT:**Bidirectional Encoder Representations from Transformers

Chapter 1

Introduction

GNN stands for Graph Neural Networks. Graphs are used to represent complex systems such as social networks, chemical compounds, and physical systems. GNNs are a class of neural networks designed to operate on graphs, meaning that they take a graph as input and produce a graph as output.

The main idea behind GNNs is to learn node embeddings, which are representations of the nodes in the graph that capture the graph's structural and semantic information. GNNs achieve this by performing message passing between neighboring nodes, where each node aggregates information from its neighbors and updates its own representation. This process is repeated for multiple rounds, allowing the node embeddings to capture increasingly complex information about the graph.

GNNs have been successful in a wide range of applications, including node classification, link prediction, graph classification, and recommendation systems. They have also been extended to handle different types of graphs, such as directed graphs and hypergraphs, and to incorporate different types of information, such as node attributes and edge weights.

Overall, GNNs represent an exciting and rapidly growing field of research, with many promising applications across various domains.

Chapter 2

Background Study

2.1 Paper 1

Node Classification in Social Networks: Smriti Bhagat, Graham Cormode,S. MuthuKrishnan

- Explores Graph Labeling Problem Given a social network (namely Twitter), with labels on some nodes, to provide a high-quality labeling for every node.
- Phenomenons of social networks discussed: homophily and co-citation
- Methods using Local Classifiers and Iterative Classification Method explored
- Drawbacks: Some methods do not scale up to large graphs

2.2 Paper 2

Scalable Learning of Users' Preferences Using Networked Data: Jiliang Tang,Huan Liu, Ali Abbasi

- Social influence theory indicates that a user's preference is influenced by influential users in her social networks and this local pattern suggests that an influential user and users influenced by this user should share similar preferences
- Weighted-vote relational neighbor (wvRN)
- Proposed Scalable Algorithm LSOCDIM

2.3 Paper 3

Predicting the Political Alignment of Twitter Users: Michael D. Conover,Bruno Goncalves Jacob Ratkiewicz,Alessandro Flammini, Filippo Menczer

- Using a data set of 1,000 manually-annotated individuals, we find that a support vector machine (SVM) trained on hash tag metadata has an accuracy on 91
- For content-based classifications they use linear support vector machines (SVMs) to discriminate between users in the 'left' and 'right' classes.
-

2.4 Paper 4

Quantifying controversy on Social Media: Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, Michael Mathioudakis

- Represent topic using a conversation graph. In such graphs, vertices represent users and edges represent conversation activity and interactions, such as posts, comments, mentions or endorsement.
- Feature selection techniques include Follow graph, Retweet Graph, Content Graph, Hybrid Content and Retweet Graph.

Chapter 3

REQUIREMENT ANALYSIS

3.1 Software requirements:

- Anaconda environment
- Jupyter Notebook
- TensorFlow
- Chrome browser

3.2 Hardware requirements:

- OS: Windows 7 or above (64-bit version)
- RAM: min 8 GB
- 2 GB min space
- Min 2 GB GPU

3.3 Language used:

- Python

Chapter 4

DETAILED DESIGN

The methodology for analyzing public opinion incorporates (1) pre-processing (2) feature extraction. (3) graph building (4) visualize data (5) Node classification

4.1 Steps Involved:

4.1.1 Data collection:

Dataset was provided by our mentor.

4.1.2 Data Pre-Processing

After getting the dataset that contains tweets around the Indian Election, the next step was to clean the data to provide the input for the text classification model. Accuracy of feature extraction also greatly depends on the quality of text data. The following are the steps performed for data cleaning.

4.1.3 Normalization

This refers to the conversion of any non-text information into the textual equivalent. For this, we have used a library called normalize. This library is based on the nltk package, so it expects nltk word tokens.

4.1.4 Removal of Punctuation marks and symbols

A regular expression is used to eliminate the unnecessary punctuation marks(,;!'".?/* etc) and symbols like emojis, emoticons. removed. URLs, extra line feeds.

4.1.5 Tokenization and Removal of Stop Words

- Tokenize: This breaks up the strings into a list of words or pieces based on a specified pattern using Regular Expressions.
- Stop Words: Stop words are generally the most common words (such as "the", "a", "an", "in") in a language. These words are of no use because they don't help us to find the context or the true meaning of a sentence. We would not want these words to take up space in our database, or taking up the valuable processing time. These words were removed from the previously cleaned tweet text using a famous NLTK package nltk.stopwords.

4.2 Libraries for Language Detection

4.2.1 Spacy

- Spacy is one of the libraries that can detect a language in text and store it
- The objective was to store the language and then filter it through the data frame
- Drawback: This language returns the detected language of longer sentences

4.2.2 Pycl2

- Pycld2 can detect languages in all sentences.
- Drawback: It detects the languages but its use was nullified since English was used in pieces and joining them would make the tweet irrelevant.

4.2.3 TextBlob

- TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.
- Drawback: This library detects only a single language

4.2.4 Defining a custom Ascii function

- Applying the ascii function on the data frame eliminated any character that is not recognised in ascii.
- This resulted in keeping only English alphabets, numbers and special characters, hence proving to be the most efficient method for language detection

4.3 MODELS USED TO GENERATE WORD EMBEDDING

It's a representation of text where words that have the same meaning have a similar representation. In other words, it represents words in a coordinate system where related words, based on a corpus of relationships, are placed closer together. In the deep learning frameworks such as TensorFlow, Keras, this part is usually handled by an embedding layer which stores a lookup table to map the words represented by numeric indexes to their dense vector representations.

4.3.1 WORD2VEC

Gensim implementation of the word2vec model is used for the training of the dataset. The first step is to prepare the text corpus for learning the embedding by creating word tokens, removing punctuation, removing stop words, etc. This is done by the Tokenizer function of Keras. The word vector thus generated is passed to the gensim.models.

Word2Vec function which trains the word embedding on the dataset. The model then uses this pre-trained word2vec embedding for the classification of tweets.

4.3.2 GLoVE

GloVe stands for global vectors for word representation. It is an unsupervised learning algorithm for generating word embeddings by aggregating a global word-word co-occurrence matrix from a corpus. The resulting embeddings show interesting linear substructures of the word in vector space.

4.3.3 BERT

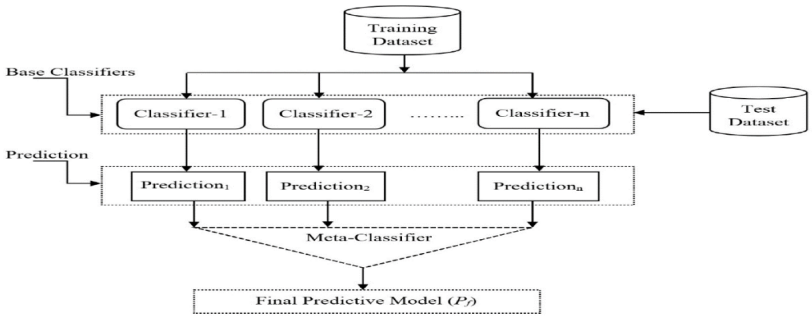
BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

Chapter 5

IMPLEMENTATION

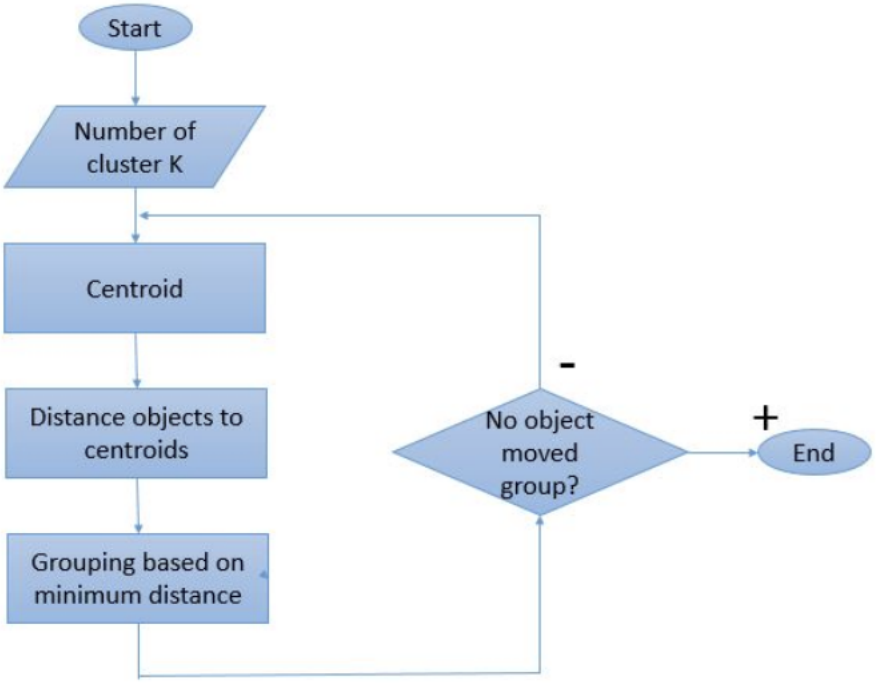
5.1 Classification Using Base Classifiers.

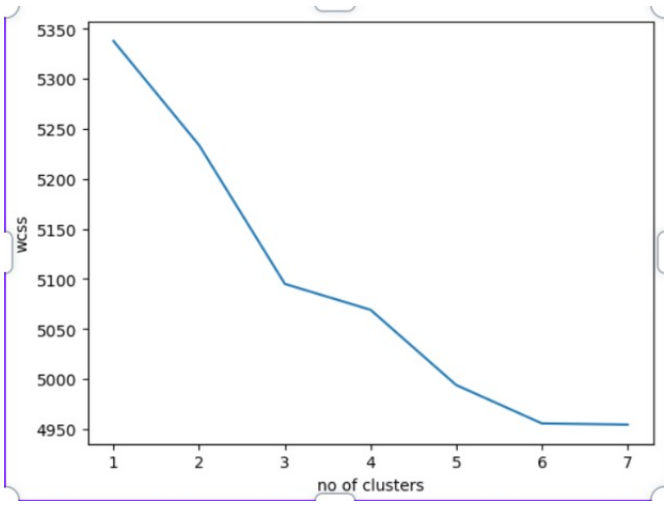
A classifier in machine learning is an algorithm that automatically orders or categorizes data into one or more of a set of "classes." One of the most common examples is an email classifier that scans emails to filter them by class label: Spam or Not Spam.



5.2 K means Clustering

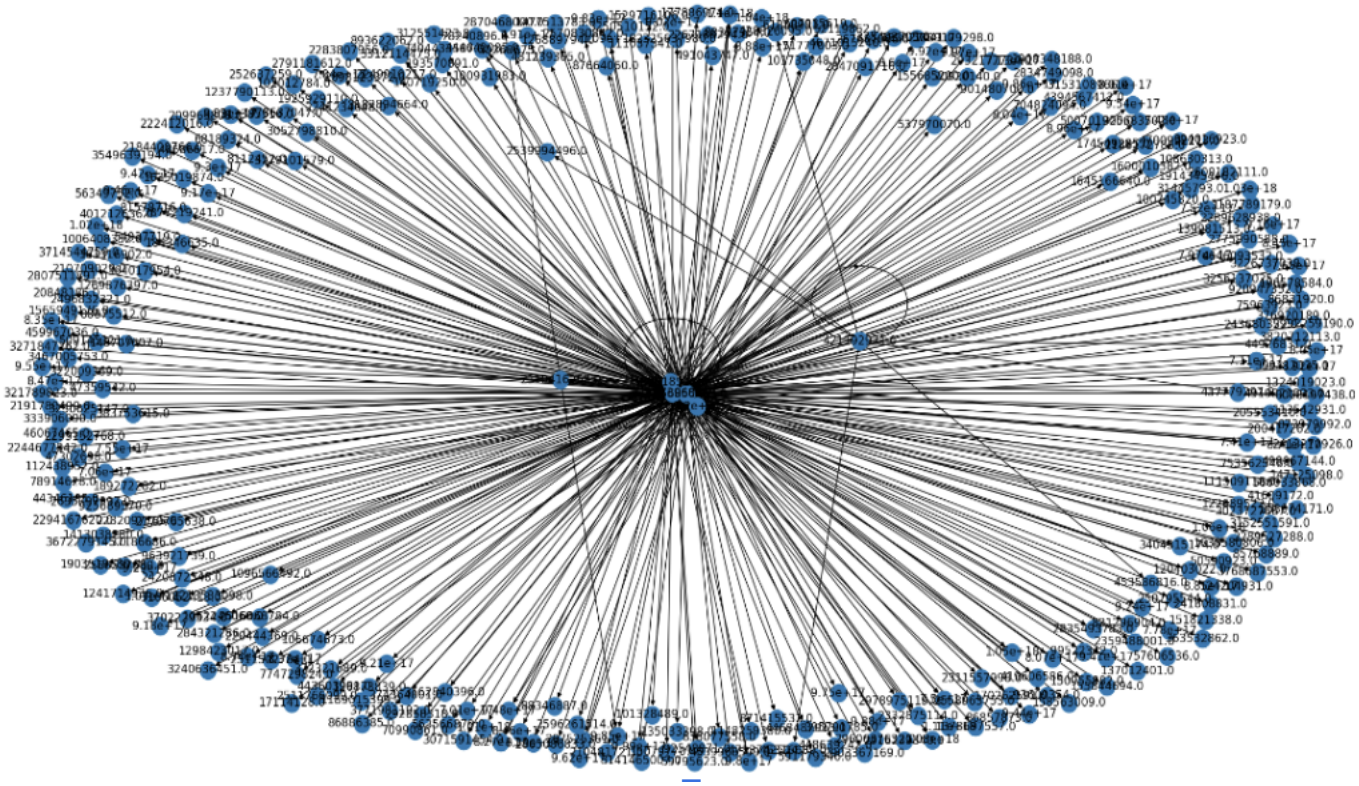
- K Means Clustering can separate data points into different clusters
- SVMs were not used since they can classify data in two classes only.
- We have used word vectorization to find similarity within words and collectively, similar sentences are grouped in a similar cluster





5.3 Graph Building using Networkx

NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and function of complex networks. It is used to study large complex networks represented in form of graphs with nodes and edges. Using networkx we can load and store complex networks. We can generate many types of random and classic networks, analyze network structure, build network models, design new network algorithms and draw networks.



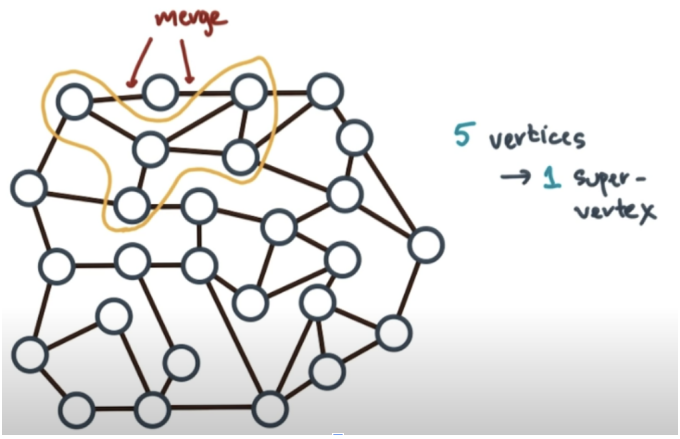
5.4 Graph Partitioning

METIS:

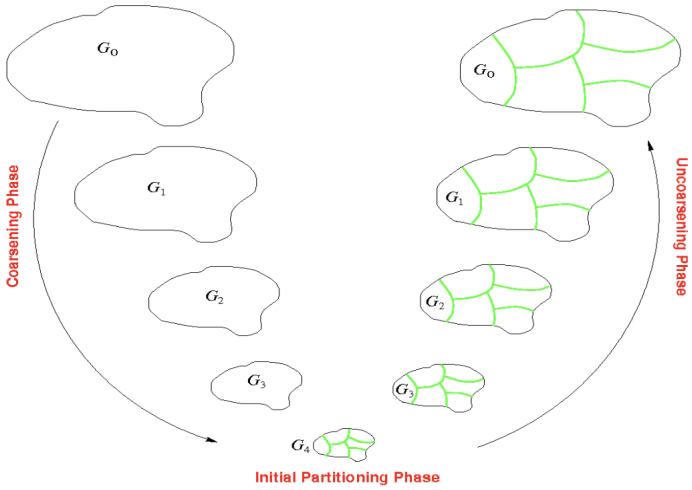
- **Coarsening:** The first step in the METIS algorithm is to coarsen the graph by grouping nodes together to create a smaller, coarser graph. This is done by repeatedly collapsing pairs of nodes into a single node until the graph has been reduced to a manageable size.

- **Partitioning:** Once the coarser graph has been generated, METIS partitions it into multiple subgraphs using a partitioning algorithm such as k-way partitioning or recursive bisection. The goal of this step is to ensure that each subgraph has roughly the same number of nodes.

- **Refinement:** After the initial partitioning, METIS refines the partition by moving nodes between subgraphs to improve the balance of the subgraphs. This is done using techniques such as Kernighan-Lin (KL) refinement or Fiduccia-Mattheyses (FM) refinement.



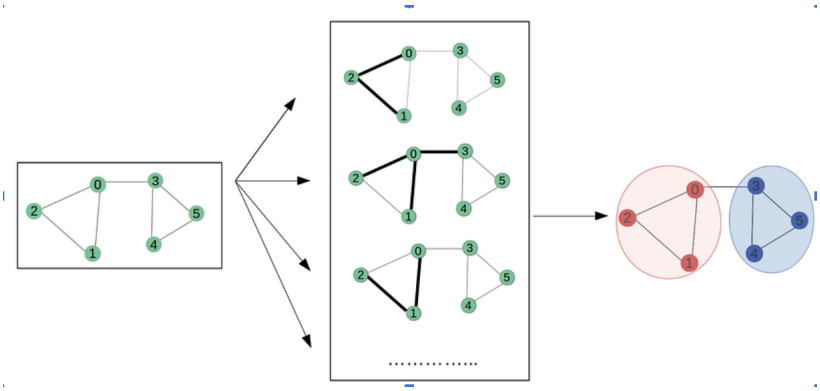
- **Un-Coarsening:** Once the partition has been refined, METIS uncoarsens the graph by recursively expanding the coarser graphs back to the original size, while maintaining the partitioning structure that was generated in the previous steps.
- **Final refinement:** Finally, METIS performs a final refinement of the partition by applying refinement techniques to the entire graph. This helps to improve the balance of the subgraphs and minimize the number of edges that cross between different subgraphs.



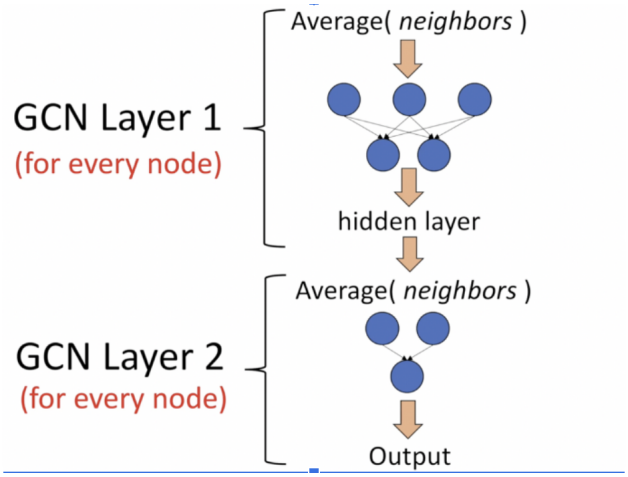
5.5 Graph Convolutional Networks

GCN is a type of convolutional neural network that can work directly on graphs and take advantage of their structural information. it solves the problem of classifying

nodes (such as documents) in a graph (such as a citation network), where labels are only available for a small subset of nodes (semi-supervised learning)



The general idea of GCN: For each node, we get the feature information from all its neighbors and of course, the feature of itself. Assume we use the average() function. We will do the same for all the nodes. Finally, we feed these average values into a neural network. In practice, we can use more sophisticated aggregate functions rather than the average function. We can also stack more layers on top of each other to get a deeper GCN. The output of a layer will be treated as the input for the next layer.



Chapter 6

FUTURE SCOPE

The future of sentiment analysis is going to continue to dig deeper, far past the surface of the number of likes, comments and shares, and aim to reach, and truly understand, the significance of social media interactions and what they tell us about the consumers behind the screens.

As a result of deeper and better understanding of the feelings, emotions and sentiments of a brand or organization's key, high-value audiences, members of these audiences will increasingly receive experiences and messages that are personalized and directly related to their wants and needs.

Again, sentiment analysis is on the verge of breaking into new areas of application. While we will likely always think of it first in terms of the traditional marketing sense, the world has already seen a few ways that sentiment analysis can be used in other areas.

Chapter 7

CONCLUSION

Algorithms have long been at the foundation of most forms of analytics, including social media and sentiment analysis. With recent years bringing big leaps in machine learning and artificial intelligence, many analytics solutions are looking to these technologies to replace algorithms. Unfortunately for organizations looking to leverage sentiment analysis to measure audience emotions, machine learning isn't yet ready to tackle the complex nuances of text and how we talk, especially on social media channels that are rife with slang, sarcasm, double meanings and misspellings. These make it difficult for artificial intelligence systems to accurately sort and classify sentiments on social media. And, with any analysis project, accuracy is crucial. It is uncertain if machine learning will progress to the point that it is capable of accurately analyzing text, or if sentiment analysis projects will have to find a new basis to avoid the current plateau of algorithms.

Chapter 8

Refrences

- Multi-Label Emotion Classification on Code-Mixed Text: Data and Methods
- Emotion Detection in Online Social Networks: A Multi-Label Learning Approach
- https://analyticsindiamag.com/the-continuous-bag-of-words-cbow-model-in-nlp-hands-on-implementation-with-codes/
- https://towardsdatascience.com/sentiment-analysis-using-lstm-and-glove-embeddings-99223a87fe8e
- www.youtube.com
- https://github.com/facebookresearch/fastText/tree/master/python