Emotion Analysis

PROJECT REPORT

BY

Saquib Sattar

28novsaquib@gmail.com

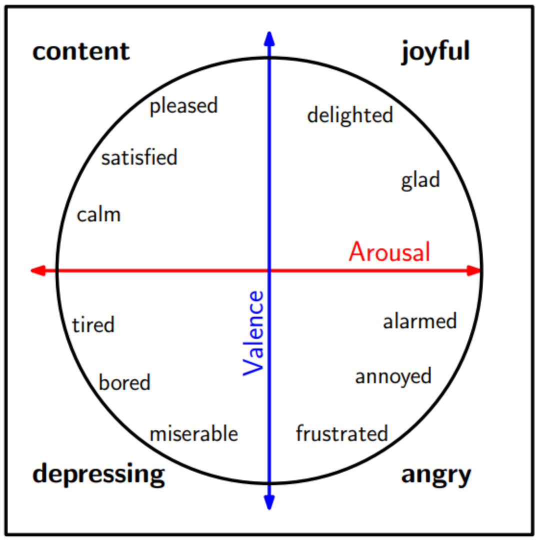
https://www.linkedin.com/in/saquib42/

https://github.com/Saquio

Abstract

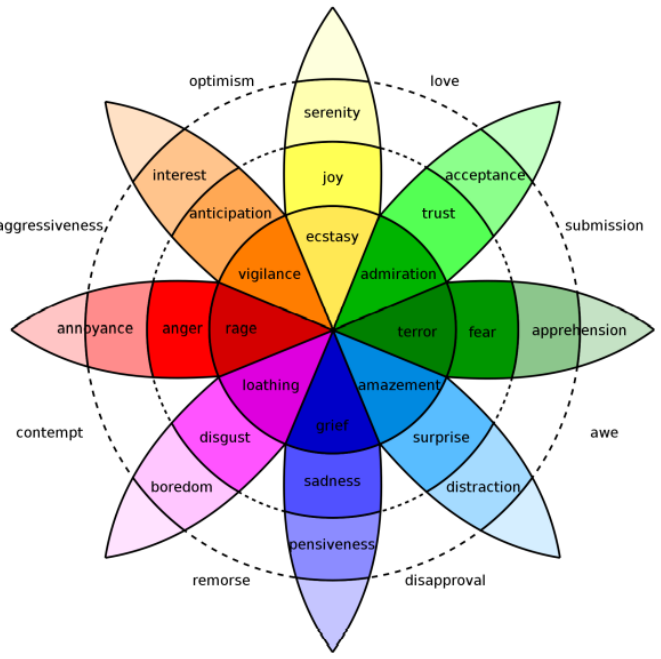
Analyzing emotion using computational techniques is one of the prominent areas of research these days. Both research domains, emotion analysis and deep learning, are promising technologies of AI which has widely been used to solve complex real-life problems. In this modern era of E-commerce, decisions are immensely dependent on sharing and posting opinions, reviews, and emotions on world wide web. Text classification was performed manually, and engineering tasks applied on the text were accomplished using handcrafted features. This was achieved by labeling the text through some predefined knowledge-based techniques, taking use of dictionaries, or using ontologies by joining hierarchical components seen in texts using graph data structures. Humans mainly did all these tasks with no automation. In the fast-developing phase, need is to develop some automatic procedure of classifying the text with least or no intervention of humans, taking machines using artificial intelligence techniques and algorithms for such kind of feature engineering tasks. A deep neural network as compared to the traditional neural network can have hundreds of layers to get more accurate results. It is the combination of various processing layers controlled by various biological nervous systems. There is basically one input layer, multiple hidden layers, and one output layer. All these layers relate to each other through neurons, where the output of one hidden layer becomes the input for other layer. The multilayer concept of deep learning gives fruitful results especially in the field of speech and image recognition. Various neural networks architectures exist for deep learning: Multilayer perceptrons, which are the oldest and simplest one's Convolutional neural networks (CNN), especially for image processing and text classification. This chapter will propose the novel method of classifying the opinions by automatically training the classifier. The details of the layers and other parameters will be discussed in the chapter highlighting on the learning is achieved through word representations. The accuracy achieved using several information retrieval metrics will be illustrated using visualization tools.

Basic Emotions



The aim was to predict whether the emotion is sad, happy, angry or fear from text data. However, we have variety of emotions.

The creation of social media has provided a platform for people to publicly share their views. Emotion analysis is also referred to as opinion mining; however, there is little difference between these two: the opinion mining extracts and analyzes people's views about an entity while emotion analysis searches for the emotion words/expression in a text and then analyzes it. It is a field that encompasses natural language processing that constructs systems that try to find out and extract different opinions within the text.



Sentiment Analysis vs Emotion Analysis

Sentiment can be typically positive/negative/neutral. Sentiment Analysis can be applied to product reviews, to identify if the reviewer liked the product or not. This has got nothing to do with emotions as such.

Emotion recognition would typically work on conversational data (e.g., from conversations with chatbots), and it would attempt to recognize the emotional state of the user angry/happy/sad...However, there can be some same overlap: if the user is happy, they will typically express positive sentiments on something.

Also: emotion recognition goes beyond text (e.g., facial expressions), whereas sentiment analysis mostly works with textual data only.

Classification techniques

Emotion analysis deals with two approaches related to classification.

Supervised approach
In this approach, the classifier is trained according to the labeled examples which are similar to test examples.

Unsupervised approach
In this approach, labels are assigned on basis of internal differences among data points only.

Classification types
Machine learning algorithms These algorithms deal with the artificial intelligence branch. It is used to build those models that have the capability to learn from given data. The supervised approach learns to do mapping between inputs to expected targets. These algorithms must be able to do a generalization of training data after the proper implementation of the training procedure. It can map new data accurately.

Naive Bayes
This is a simple as well as a probabilistic model that depends on the feature independent's assumption to classify the given data. This approach is useful for text classification also. It is very simple to implement, low cost, and has high accuracy. In this algorithm, every word in the given training set is considered and then the probability of the word in each class is calculated. Then this algorithm gets ready for classifying new data. The new sentence is then classified and spitted into a single word. The advantage provided by this algorithm is that it makes full utilization of all the available pieces of evidence to make a classification.

Support vector machine
This is the non-probabilistic model that works on the principle of plotting training data into the multidimensional space and then separate classes along the hyperplane. If the classes are not linearly separable, then the algorithm would add on a new dimension to separate different classes. The major disadvantage of this approach is that size of feature space increases due to the addition of extra dimensions to the feature space

Decision tree
This is the most widely used algorithms that can be applied to any kind of data. It segregates training data into smaller portions to identify patterns for applying classification. After this, the knowledge is depicted in a logical structure like flowcharts.

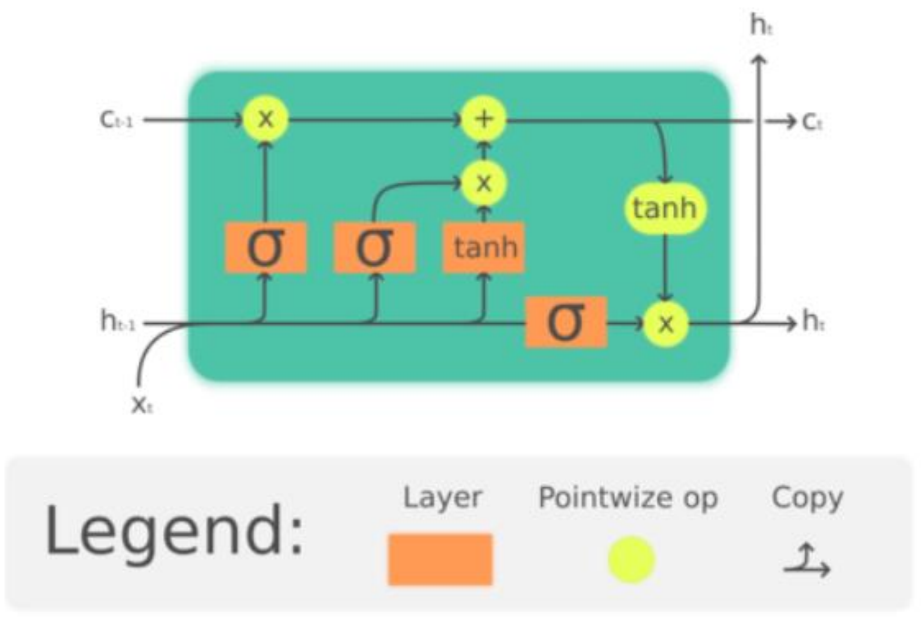
Deep neural networks

The idea behind deep neural networks is not new but dates back to the 1950s. However, it became possible to practically implement it because of the high-end resource capability available nowadays. Deep artificial neural networks have given outstanding performance in the field of sentiment analysis. They have the capability of designing modeling and processing nonlinear relationships. Most deep learning methods make use of neural networks.

There are various deep learning neural networks available. Some of them are as follows. 9.8.1 Deep belief network This is one of the deep neural networks that works on the principle of multilayer belief to learn a layer of features from visible units using the contrastive divergence algorithm. It treats activations of previously trained features as visible units and then learns features of features. Finally, the whole NN is trained when the learning for the final hidden layer is achieved.

LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems)



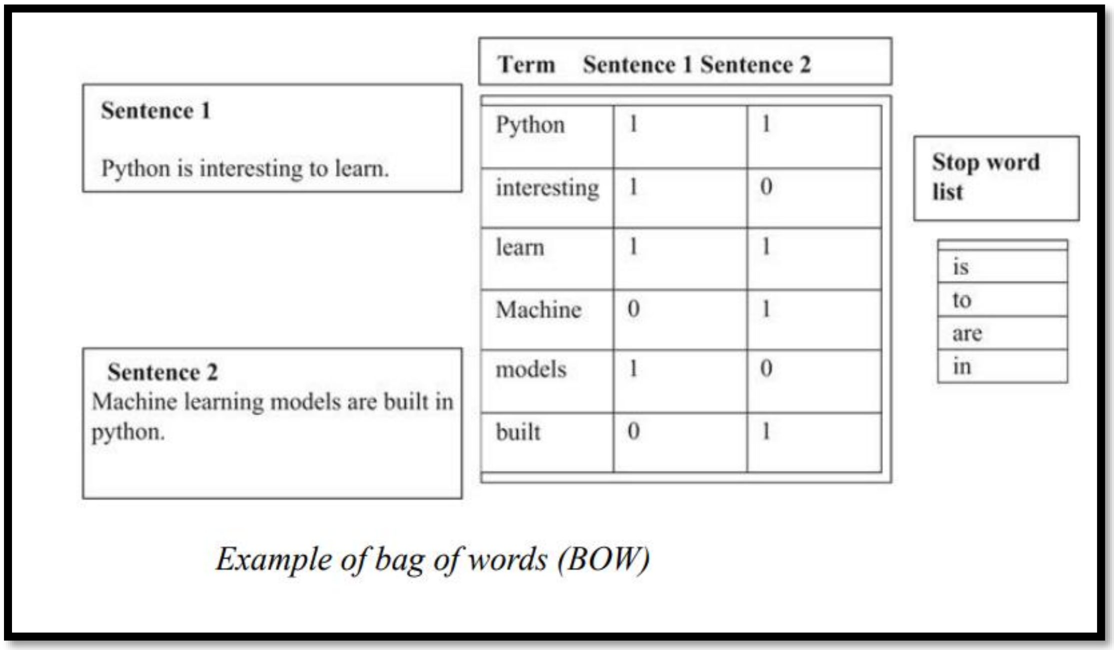
Bag of words (BOW)

Generally, a sparse bag of words representation is used for documents or sentences. The text contains words (broken into tokens). For example, two sentences as shown below:

Sentence 1: Python is interesting to learn.

Sentence 2: Machine learning models are built in python.

Weighting features need to be integrated with bag of words as not all the words should be taken for consideration. Figure shows the matrix constructed using the above example.



Sentence 1

Python is interesting to learn.

Sentence 2

Machine learning models are built in python.

Term	Sentence 1	Sentence 2
Python	1	1
interesting	1	0
learn	1	1
Machine	0	1
models	1	0
built	0	1

Stop word list
is
to
are
in

Example of bag of words (BOW)

TF-IDF

TF-IDF stands for *term frequency-inverse document frequency*, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

One of the simplest ranking functions is computed by summing the tf-idf for each query term; many more sophisticated ranking functions are variants of this simple model.

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However, it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

IDF(t) = log_e(Total number of documents / Number of documents with term t in it)

Converting text into numbers

Our labels are in numerical form (0 and 1) but our Text are in string form. A machine learning algorithm requires its inputs to be in numerical form.

In NLP, there are two main concepts for turning text into numbers:

- **Tokenization** - A straight mapping from word or character or sub-word to a numerical value. There are three main levels of tokenization:
    1. Using **word-level tokenization** with the sentence "I love TensorFlow" might result in "I" being 0, "love" being 1 and "TensorFlow" being 2. In this case, every word in a sequence considered a single token.
    2. **Character-level tokenization**, such as converting the letters A-Z to values 1-26. In this case, every character in a sequence considered a single token.
    3. **Sub-word tokenization** is in between word-level and character-level tokenization. It involves breaking individual words into smaller parts and then converting those smaller parts into numbers. For example, "my favorite food is pineapple pizza" might become "my, fav, avour, rite, fo, oo, od, is, pin, ine, app, le, piz, za". After doing this, these sub-words would then be mapped to a numerical value. In this case, every word could be considered multiple tokens.

- **Embeddings** - An embedding is a representation of natural language which can be learned. Representation comes in the form of a feature vector. For example, the word "dance" could be represented by the 5-dimensional vector [-0.8547, 0.4559, -0.3332, 0.9877, 0.1112]. It's important to note here, the size of the feature vector is tunable. There are two ways to use embeddings:
    1. **Create your own embedding** - Once your text has been turned into numbers (required for an embedding), you can put them through an embedding layer (such as tf.keras.layers.Embedding) and an embedding representation will be learned during model training.
    2. **Reuse a pre-learned embedding** - Many pre-trained embeddings exist online. These pre-trained embeddings have often been learned on large corpuses of text (such as all of Wikipedia) and thus have a good underlying representation of natural language. You can use a pre-trained embedding to initialize your model and fine-tune it to your own specific task.

Final Result

Model	**Accuracy**	**Type**
Logistic Regression	81.97	Machine Learning
Naive Bayes	73.78	Machine Learning
Support Vector Classifier	83.84	Machine Learning
Random Forest Classifier	82.30	Machine Learning
LSTM	77.86	Deep Learning