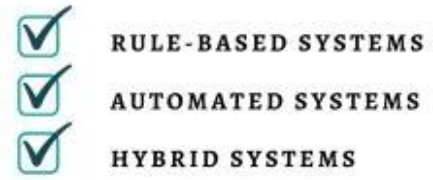
WEEK 12 – Session 2

NLP use case - Sentiment analysis (SA)

How does sentiment analysis work?

There are different algorithms that you can use to perform sentiment analysis



- ✓ **Rule-based Approaches**

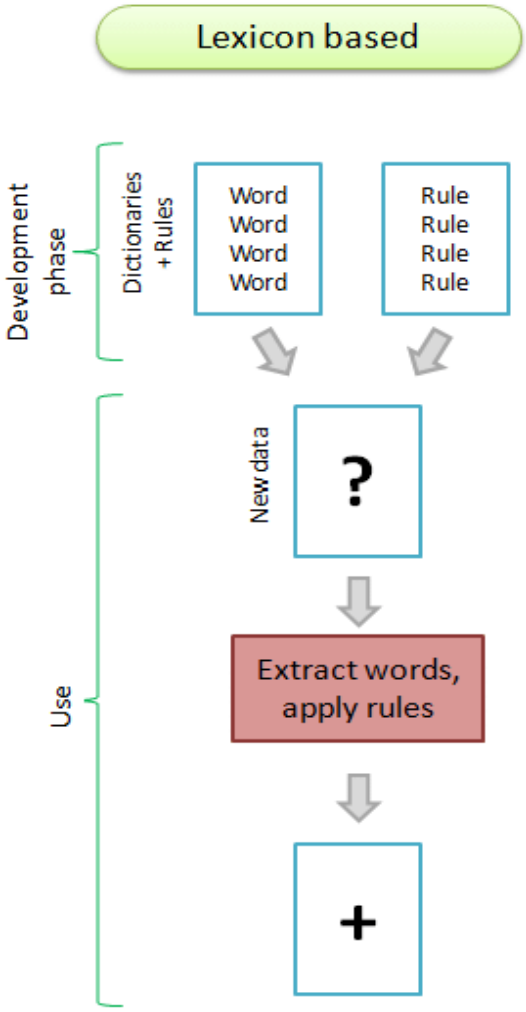
  Usually, a rule-based system uses a set of human-crafted rules to help identify subjectivity, polarity, or the subject of an opinion.

  These rules may include various NLP techniques developed in computational linguistics, such as:

  - Stemming, tokenization, part-of-speech tagging and parsing.
  - Lexicons (i.e. lists of words and expressions).

Here's a basic example of how a rule-based system works:

1. Defines two lists of polarized words (e.g. negative words such as bad, worst, ugly, etc and positive words such as good, best, beautiful, etc).
2. Counts the number of positive and negative words that appear in a given text.
3. If the number of positive word appearances is greater than the number of negative word appearances, the system returns a positive sentiment, and vice versa. If the numbers are even, the system will return a neutral sentiment.

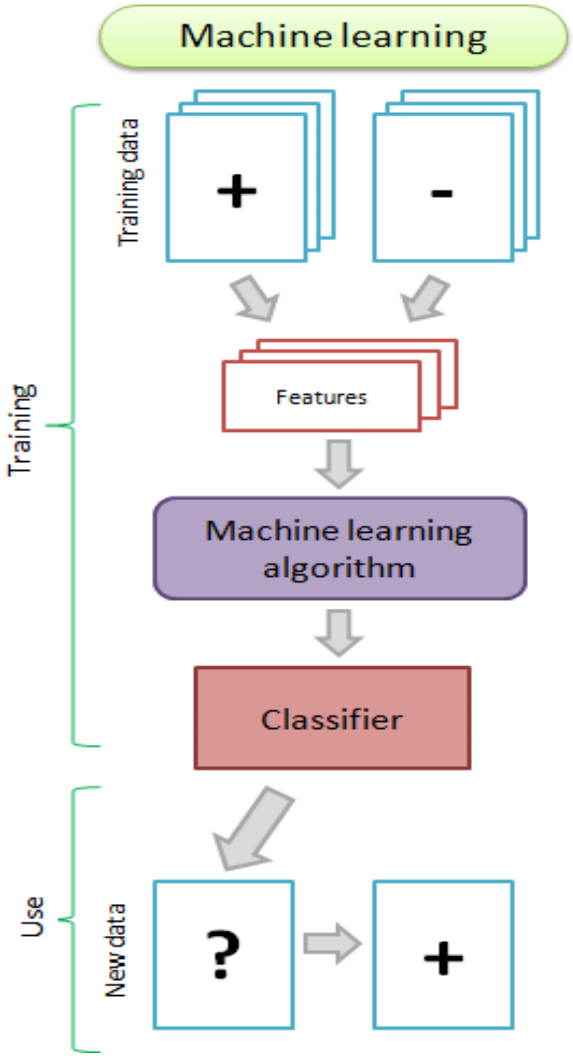


Rule-based systems are very naive since they don't take into account how words are combined in a sequence. Of course, more advanced processing techniques can be used, and new rules added to support new expressions and vocabulary. However, adding new rules may affect previous results, and the whole system can get very complex. Since rule-based systems often require fine-tuning and maintenance, they'll also need regular investments.

✓ **Automatic Approaches**

Automatic methods, contrary to rule-based systems, don't rely on manually crafted rules, but on machine learning techniques. A sentiment analysis task is usually modeled as a classification problem, whereby a classifier is fed a text and returns a category, e.g. positive, negative, or neutral.

Here's how a machine learning classifier can be implemented:



a] Feature Extraction from Input Text

Machine learning text classifiers will transform the text extraction using the classical approach of bag-of-words or bag-of-n-grams with their frequency. A new feature extraction system is created on word embeddings known as word vectors.

This kind of representation helps to improve the performance of classifiers by making it possible for words with similar meanings to have similar presentations

b] Training and Prediction

In the training process, your model links with a particular input(i.e., text) to the corresponding output based on the test sample. The feature extractor will help to transfer the input to the feature vector. These pairs of feature vectors and the tags provided are transferred to the machine learning algorithm to generate a model.

In the prediction process, the feature extractor transforms the unidentified text inputs into feature vectors. Further, these feature vectors generate the predicted tags like positive, negative, and neutral.

Classification Algorithms

Various classification algorithms involve statistical modelings like naive Bayes, support vector machines, deep learning, or logistic regression. Let us discuss them in detail below:

- **Naive Bayes:** It is a family of probabilistic algorithms that predict the category of a text by using the Bayes theorem.
- **Support Vector Machines:** It is a non-probabilistic model that uses a representation of the input text as a point in multi-dimensional space. Different text categories map to distinct regions within the space because the new texts are categorized based on the similarity with the existing text and the region they are mapping.

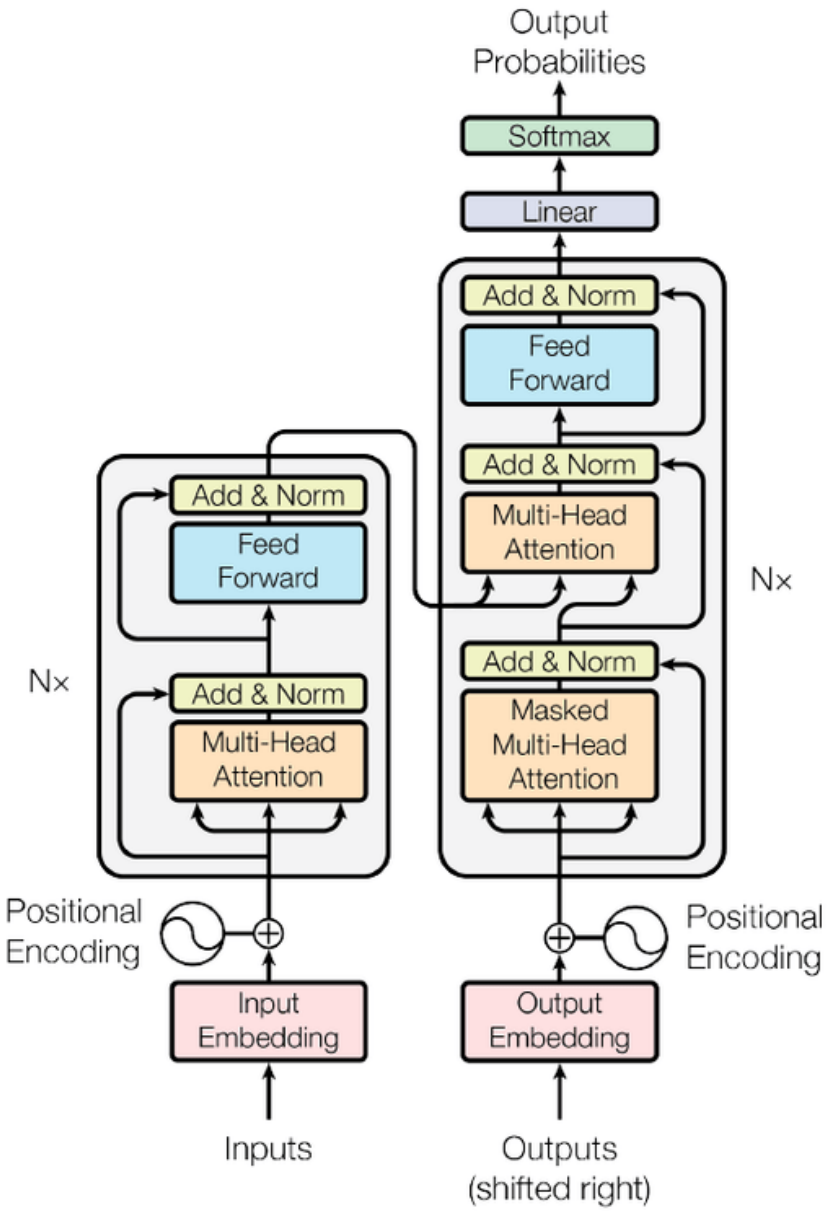
- **Deep Learning:** A family of algorithms that attempts to mimic the human brain with the help of artificial neural networks to process the data.
- **Linear Regression:** A family of algorithms in statistics that helps to predict some value (y) for a given set of features (x).

✓ **Hybrid Sentiment Analysis Algorithms**

The hybrid model is the combination of elements of the rule-based approach and automatic approach into one system. A massive advantage of this approach is that the results are often more accurate and precise than the rule-based and automated approaches.

NLP's Transformer

NLP's Transformer is a new architecture that aims to solve tasks sequence-to-sequence while easily handling long-distance dependencies. Computing the input and output representations without using sequence-aligned RNNs or convolutions and it relies entirely on self-attention. Lets look in detail what are transformers. The Basic Architecture:

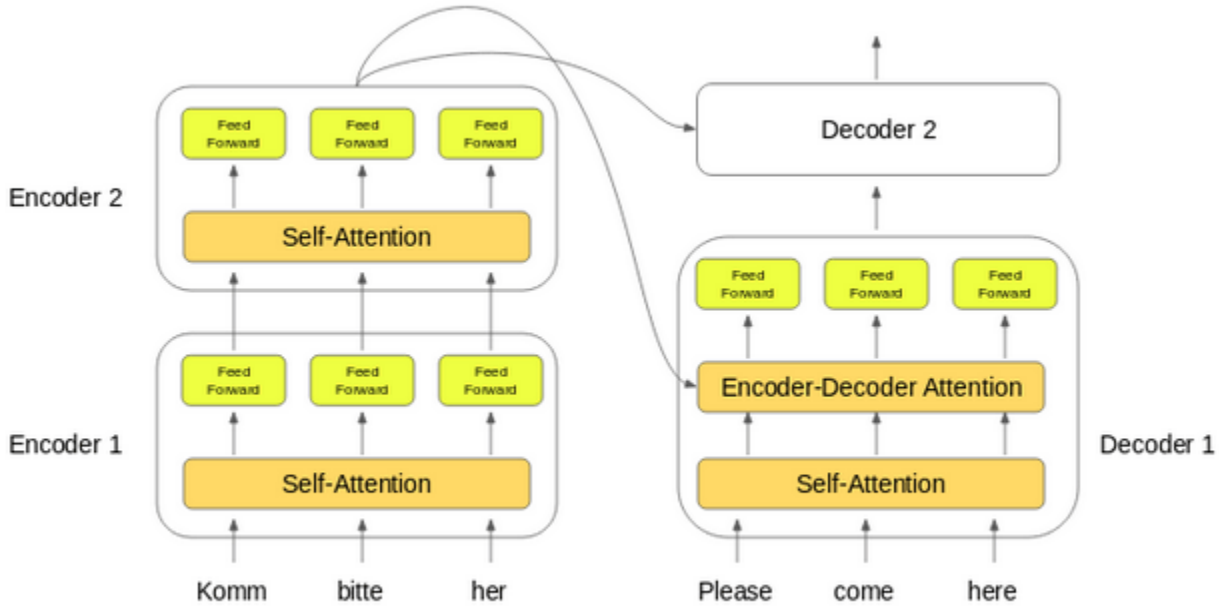


In general, the Transformer model is based on the encoder-decoder architecture. The encoder is the gray rectangle on the left and the decoder is on the right. The encoder and decoder consist of two and three sub layers, respectively. Multi-head self-awareness, fully connected feed forward network, and encoder decoder self-awareness in the case of decoders (called multi-head attention) with the following visualizations).

Encoder: The encoder is responsible for stepping through the input time steps and encoding the entire sequence into a fixed-length vector called a context vector.

Decoder: The decoder is responsible for stepping through the output time steps while reading from the context vector.

Let's see how this setup of the encoder and the decoder stack works:



- The word embeddings of the input sequence are passed to the first encoder.
- These are then transformed and propagated to the next encoder.
- The output from the last encoder in the encoder-stack is passed to all the decoders in the decoder-stack.

What exactly does this “Self-Attention” layer do in the Transformer?

Self-Attention in Transformers

Self-attention is a new spin on the attention technique. Instead of looking at prior hidden vectors when considering a word embedding, self-attention is a weighted combination of all other word embeddings (including those that appear later in the sentence):