UNIT-5

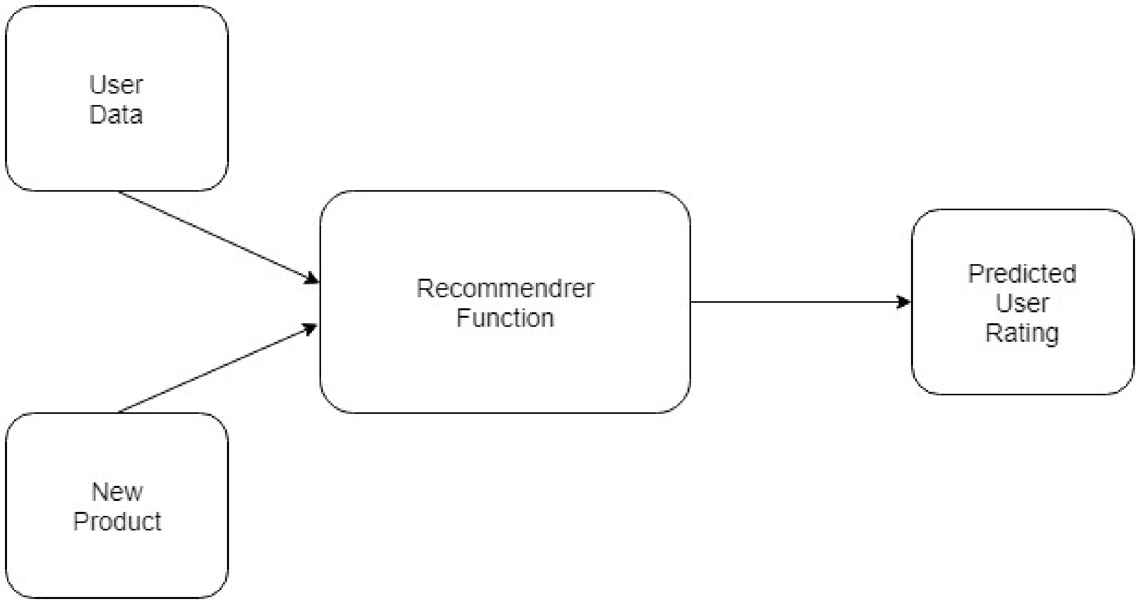
Recommender Systems:

E-commerce and retail companies are leveraging the power of data and boosting sales by implementing recommender systems on their websites. The use cases of these systems have been steadily increasing within the last years and it's a great time to dive deeper into this amazing machine learning technique.

Recommender systems aim to predict users' interests and recommend product items that quite likely are interesting for them. They are among the most powerful machine learning systems that online retailers implement in order to drive sales.

Data required for recommender systems stems from explicit user ratings after watching a movie or listening to a song, from implicit search engine queries and purchase histories, or from other knowledge about the users/items themselves.

Sites like Spotify, YouTube or Netflix use that data in order to suggest playlists, so-called **Daily mixes**, or to make **video recommendations**, respectively.



Necessity of Recommender Systems:

Companies using recommender systems focus on increasing sales as a result of very personalized offers and an enhanced customer experience.

Recommendations typically speed up searches and make it easier for users to access content they're interested in, and surprise them with offers they would have never searched for.

What is more, companies are able to gain and retain customers by sending out emails with links to new offers that meet the recipients' interests, or suggestions of films and TV shows that suit their profiles.

The user starts to feel known and understood and is more likely to buy additional products or consume more content. By knowing what a user wants, the company gains competitive advantage and the threat of losing a customer to a competitor decreases.

Providing that added value to users by including recommendations in systems and products is appealing. Furthermore, it allows companies to position ahead of their competitors and eventually increase their earnings.

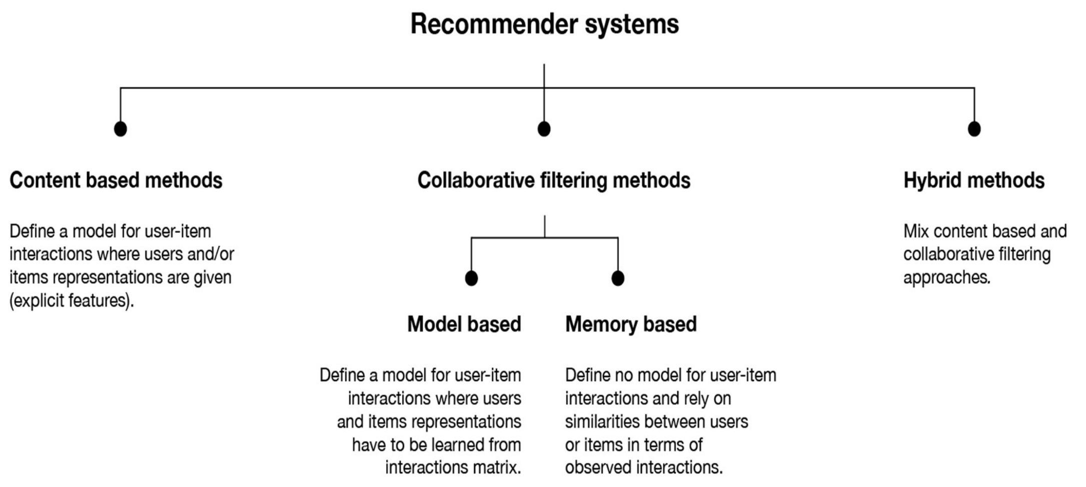
Working of Recommender Systems:

Recommender systems function with two kinds of information:

- *Characteristic information*. This is information about items (keywords, categories, etc.) and users (preferences, profiles, etc.).
- *User-item interactions*. This is information such as ratings, number of purchases, likes, etc.

Based on this, we can distinguish between three algorithms used in recommender systems:

- *Content-based* systems, which use characteristic information.
- *Collaborative filtering* systems, which are based on user-item interactions.
- *Hybrid systems*, which combine both types of information with the aim of avoiding problems that are generated when working with just one kind.


Recommender systems
Content based methods
Define a model for user-item interactions where users and/or items representations are given (explicit features).
Collaborative filtering methods
Model based
Define a model for user-item interactions where users and items representations have to be learned from interactions matrix.
Memory based
Define no model for user-item interactions and rely on similarities between users or items in terms of observed interactions.
Hybrid methods
Mix content based and collaborative filtering approaches.

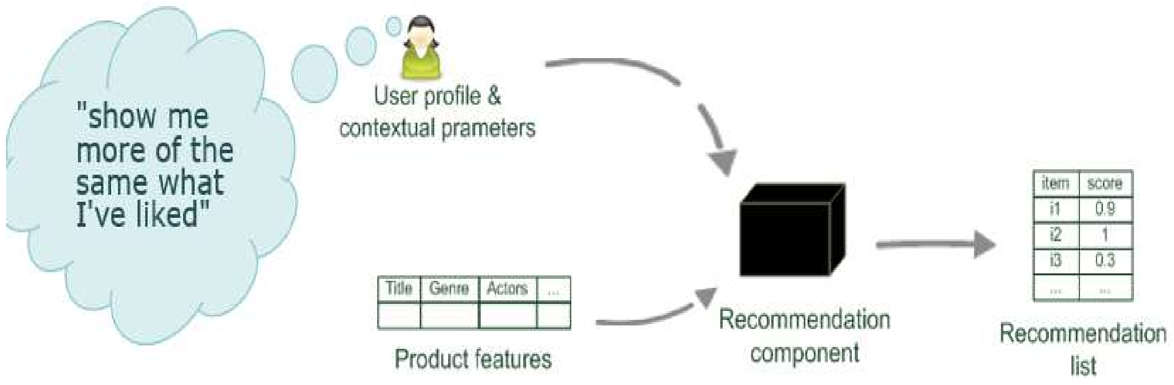
Content-based systems

These systems make recommendations using a user’s item and profile features. They hypothesize that if a user was interested in an item in the past, they will once again be interested in it in the future. Similar items are usually grouped based on their features. User profiles are constructed using historical interactions or by explicitly asking users about their interests. There are other systems, not considered purely content-based, which utilize user personal and social data.

One issue that arises is making obvious recommendations because of excessive specialization (user A is only interested in categories B, C, and D, and the system is not able to recommend items outside those categories, even though they could be interesting to them).

Another common problem is that new users lack a defined profile unless they are explicitly asked for information. Nevertheless, it is relatively simple to add new items to the system. We just need to ensure that we assign them a group according to their features.

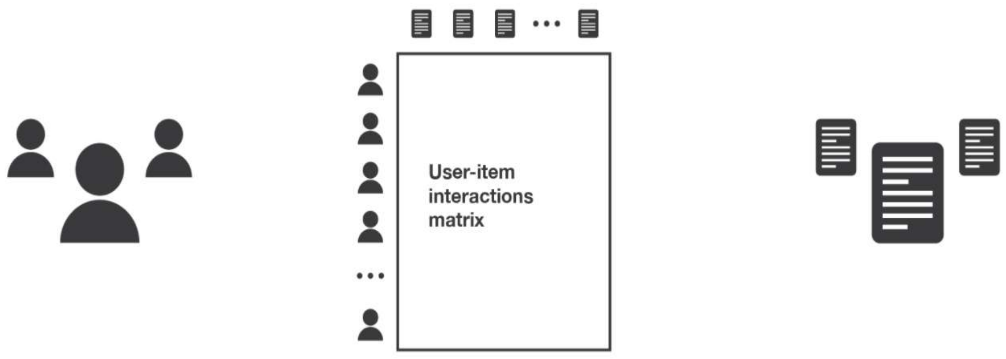
Recommender systems can also be seen as tools for dealing with information overload, as these systems aim to select the most interesting items from a larger set. Recommender systems research is also strongly rooted in the fields of information retrieval and information filtering. In these areas, however, the focus lies mainly on the problem of discriminating between relevant and irrelevant documents. Many of the techniques developed in these areas exploit information derived from the documents' contents to rank them.


"show me more of the same what I've liked"
User profile & contextual prameters
Title | Genre | Actors | ...
Product features
Recommendation component
item | score
i1 | 0.9
i2 | 1
i3 | 0.3
... | ...
Recommendation list

Collaborative filtering systems

Collaborative filtering is currently one of the most frequently used approaches and usually provides better results than content-based recommendations. Some examples of this are found in the recommendation systems of **Youtube**, **Netflix**, and **Spotify**.
These kinds of systems utilize user interactions to filter for items of interest. We can visualize the set of interactions with a matrix, where each entry (i, j)(*i*,*j*) represents the interaction between user i*i* and item j*j*. An interesting way of looking at collaborative filtering is to think of it as a generalization of **classification** and **regression**. While in these cases we aim to predict a variable that directly depends on other variables (features), in collaborative filtering there is no such distinction of feature variables and class variables.

Visualizing the problem as a matrix, we don't look to predict the values of a unique column, but rather to predict the value of any given entry.



Users	User-item interactions matrix	Items
suscribers	rating given by a user to a movie (integer)	movies
readers	time spent by a reader on an article (float)	articles
buyers	product clicked or not when suggested (boolean)	products
	...	

Illustration of the user-item interactions matrix.

In short, collaborative filtering systems are based on the assumption that if a user likes item A and another user likes the same item A as well as another item, item B, the first user could also be interested in the second item. Hence, they aim to predict new interactions based on historical ones. There are two types of methods to achieve this goal: *memory-based* and *model-based*.

Memory-based

There are two approaches: the first one identifies **clusters** of users and utilizes the interactions of one specific user to predict the interactions of other similar users. The second approach identifies clusters of items that have been rated by user A and utilizes them to predict the interaction of user A with a different but similar item B. These methods usually encounter major problems with large **sparse** matrices, since the number of user-item interactions can be too low for generating high quality clusters.

Model-based

These methods are based on machine learning and data mining techniques. The goal is to train models to be able to make predictions.

For example, we could use existing user-item interactions to train a model to predict the top-5 items that a user might like the most. One advantage of these methods is that they are able to recommend a larger number of items to a larger number of users, compared to other methods like memory-based. We say they have large *coverage*, even when working with large sparse matrices.

Issues with collaborative filtering systems

There are two main challenges that come up with these systems:

1. Cold start: we should have enough information (user-item interactions) for the system to work. If we setup a new e-commerce site, we cannot give recommendations until users have interacted with a significant number of items.
2. Adding new users/items to the system: whether it is a new user or item, we have no prior information about them since they don't have existing interactions.

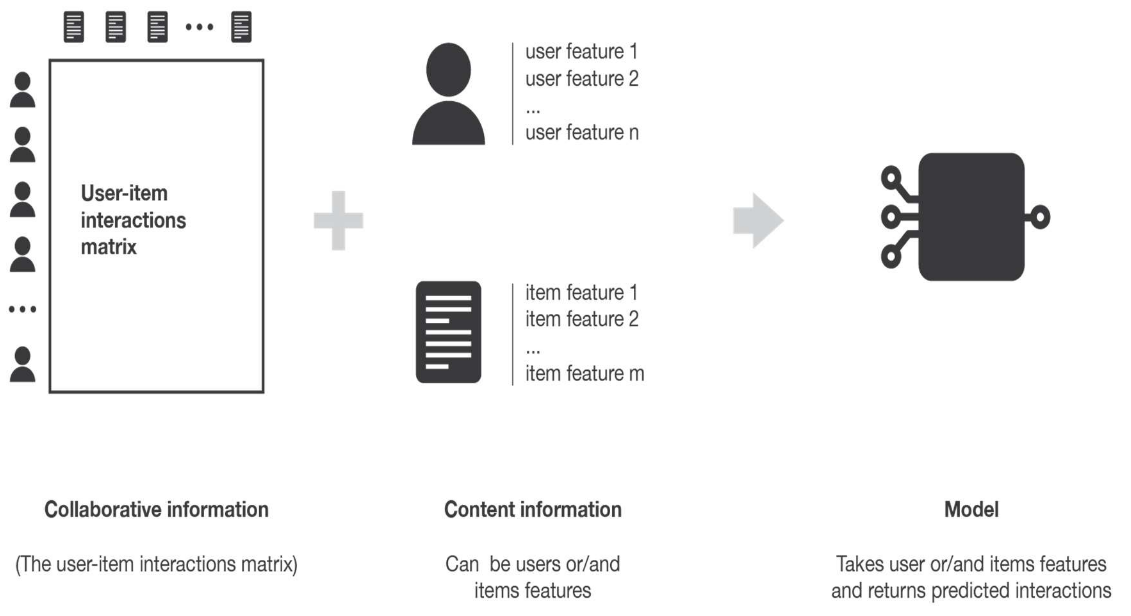
These problems can be alleviated by asking users for other type of data at the time of sign-up (gender, age, interests, etc), and using meta information from the items in order to be able to relate them to other existing items in the database.

Hybrid Approach:

This Hybrid Approach for Recommendation System is based on User Collaborative Filtering, Item-Based Collaborative Filtering and Adaptive Association Rule Mining (ARM) to improve the performance of existing Recommender systems. This recommendation system has the following major purposes associated with it:

- Avoid the problem of recommending only similar products
- Overcome the problem of false nearest neighbors
- Overcome the inefficiency of ARM algorithms to process large data
- Adaptivity in support levels required in ARM algorithms

- More personalized recommendations and specific to the user needs
- Computationally faster system than the traditional recommendation system algorithms.


User-item interactions matrix
user feature 1
user feature 2
...
user feature n
item feature 1
item feature 2
...
item feature m
Collaborative information
(The user-item interactions matrix)
Content information
Can be users or/and items features
Model
Takes user or/and items features and returns predicted interactions

Sentiment Analysis:

Sentiment analysis is a machine learning tool that analyzes texts for polarity, from positive to negative. By training machine learning tools with examples of emotions in text, machines automatically learn how to detect sentiment without human input.

To put it simply, machine learning allows computers to learn new tasks without being expressly programmed to perform them. Sentiment analysis models can be trained to read beyond mere definitions, to understand things like, context, sarcasm, and misapplied words. For example:

"Super user-friendly interface. Yeah right. An engineering degree would be helpful."

Out of context, the words 'super user-friendly' and 'helpful' could be read as positive, but this is clearly a negative comment. Using sentiment analysis, computers can automatically process text data and understand it just as a human would, saving hundreds of employee hours.

Imagine using machine learning to process customer service tickets, categorize them in order of urgency, and automatically route them to the correct department or employee. Or, to analyze thousands of product reviews and social media posts to gauge brand sentiment.

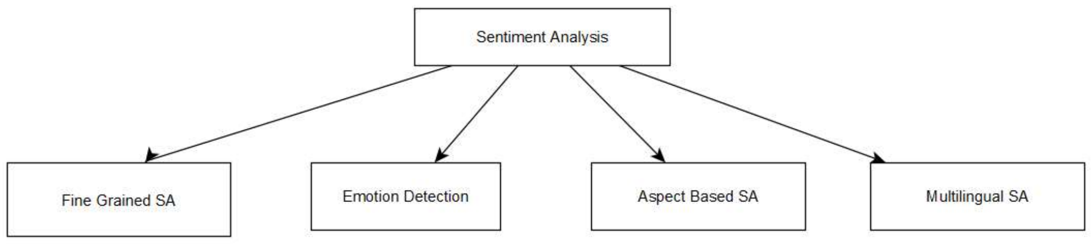
Sentiment analysis (or **opinion mining**) is a natural language processing technique used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs.

Sentiment analysis is the process of detecting positive or negative sentiment in text. It's often used by businesses to detect sentiment in social data, gauge brand reputation, and understand customers.

Since customers express their thoughts and feelings more openly than ever before, sentiment analysis is becoming an essential tool to monitor and understand that sentiment. Automatically analyzing customer feedback, such as opinions in survey responses and social media conversations, allows brands to learn what makes customers happy or frustrated, so that they can tailor products and services to meet their customers' needs.

For example, using sentiment analysis to automatically analyze 4,000+ reviews about your product could help you discover if customers are happy about your pricing plans and customer service.

Types of Sentiment Analysis:



Sentiment analysis models focus on polarity (*positive, negative, neutral*) but also on feelings and emotions (*angry, happy, sad*, etc), urgency (*urgent, not urgent*) and even intentions (*interested v. not interested*).

Depending on how you want to interpret customer feedback and queries, you can define and tailor your categories to meet your sentiment analysis needs. In the meantime, here are some of the most popular types of sentiment analysis:

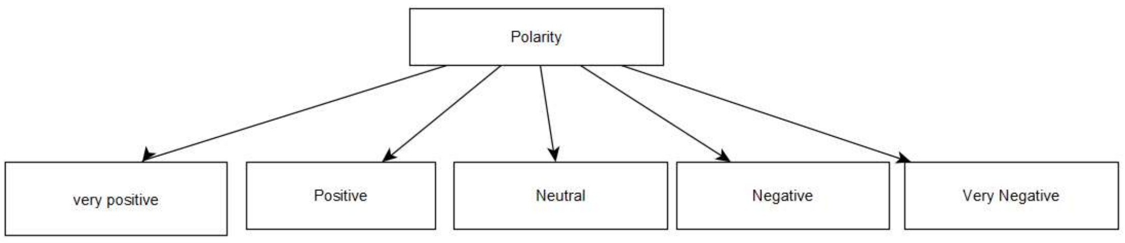
Fine-grained Sentiment Analysis

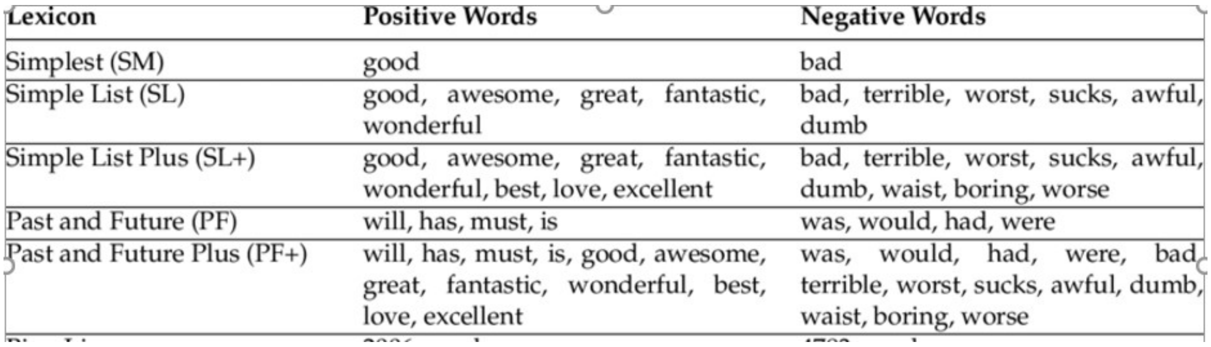
If polarity precision is important to your business, you might consider expanding your polarity categories to include:

- Very positive
- Positive
- Neutral
- Negative
- Very negative

This is usually referred to as fine-grained sentiment analysis, and could be used to interpret 5-star ratings in a review, for example:

- Very Positive = 5 stars
- Very Negative = 1 star





Lexicon	Positive Words	Negative Words
Simplest (SM)	good	bad
Simple List (SL)	good, awesome, great, fantastic, wonderful	bad, terrible, worst, sucks, awful, dumb
Simple List Plus (SL+)	good, awesome, great, fantastic, wonderful, best, love, excellent	bad, terrible, worst, sucks, awful, dumb, waist, boring, worse
Past and Future (PF)	will, has, must, is	was, would, had, were
Past and Future Plus (PF+)	will, has, must, is, good, awesome, great, fantastic, wonderful, best, love, excellent	was, would, had, were, bad, terrible, worst, sucks, awful, dumb, waist, boring, worse

Emotion detection

This type of sentiment analysis aims to detect emotions, like happiness, frustration, anger, sadness, and so on. Many emotion detection systems use lexicons (i.e. lists of words and the emotions they convey) or complex machine learning algorithms.

One of the downsides of using lexicons is that people express emotions in different ways. Some words that typically express anger, like *bad* or *kill* (e.g. *your product is so bad* or *your customer support is killing me*) might also express happiness (e.g. *this is bad ass* or *you are killing it*).

Aspect-based Sentiment Analysis

Usually, when analyzing sentiments of texts, let's say product reviews, you'll want to know which particular aspects or features people are mentioning in a positive, neutral, or negative way. That's where aspect-based sentiment analysis can help, for example in this text: *"The battery life of this camera is too short"*, an aspect-based classifier would be able to determine that the sentence expresses a negative opinion about the feature battery life.

Multilingual sentiment analysis

Multilingual sentiment analysis can be difficult. It involves a lot of preprocessing and resources. Most of these resources are available online (e.g. sentiment lexicons), while others need to be created (e.g. translated corpora or noise detection algorithms), but you'll need to know how to code to use them.

Benefits of sentiment analysis:

- **Sorting Data at Scale**

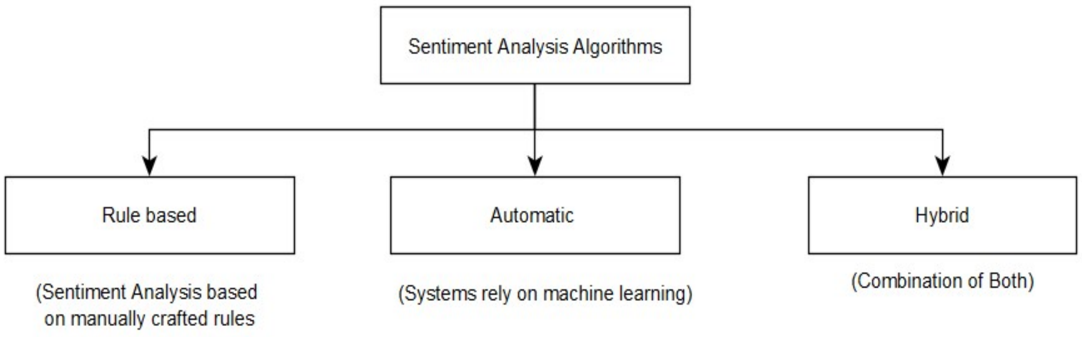
Can you imagine manually sorting through thousands of tweets, customer support conversations, or surveys? There's just too much business data to process manually. Sentiment analysis helps businesses process huge amounts of data in an efficient and cost-effective way.

- **Real-Time Analysis**

Sentiment analysis can identify critical issues in real-time, for example is a PR crisis on social media escalating? Is an angry customer about to churn? Sentiment analysis models can help you immediately identify these kinds of situations, so you can take action right away.

Sentiment analysis algorithms fall into one of three buckets:

- **Rule-based:** these systems automatically perform sentiment analysis based on a set of manually crafted rules.
- **Automatic:** systems rely on machine learning techniques to learn from data.
- **Hybrid** systems combine both rule-based and automatic


Sentiment Analysis Algorithms
Rule based
Automatic
Hybrid
(Sentiment Analysis based on manually crafted rules
(Systems rely on machine learning)
(Combination of Both)

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:

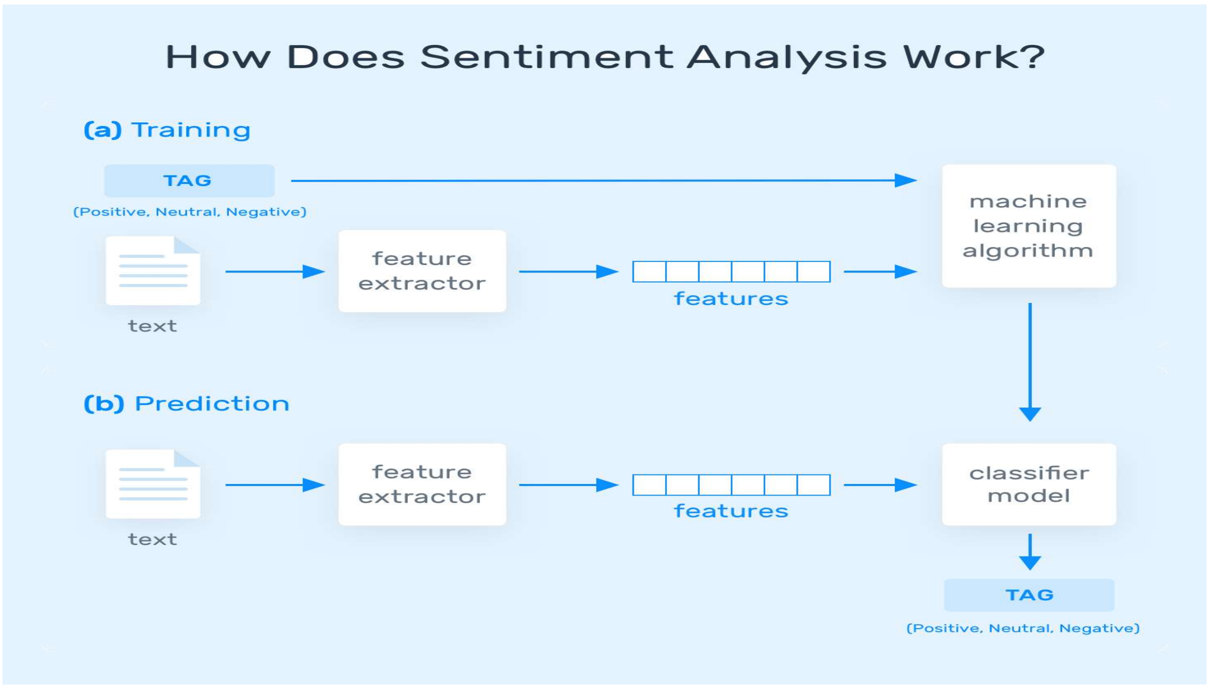
1. Defines two lists of polarized words (e.g. negative words such as *bad*, *worst*, *ugly*, etc and positive words such as *good*, *best*, *beautiful*, etc).
2. Counts the number of positive and negative words that appear in a given text.
3. If the number of positive word appearances is greater than 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:


How Does Sentiment Analysis Work?
(a) Training
TAG
(Positive, Neutral, Negative)
text
feature extractor
features
machine learning algorithm
(b) Prediction
text
feature extractor
features
classifier model
TAG
(Positive, Neutral, Negative)

The Training and Prediction Processes

In the training process (a), our model learns to associate a particular input (i.e. a text) to the corresponding output (tag) based on the test samples used for training. The feature extractor transfers the text input into a feature vector. Pairs of feature vectors and tags (e.g. *positive*, *negative*, or *neutral*) are fed into the machine learning algorithm to generate a model.

In the prediction process (b), the feature extractor is used to transform unseen text inputs into feature vectors. These feature vectors are then fed into the model, which generates predicted tags (again, *positive*, *negative*, or *neutral*).

Feature Extraction from Text

The first step in a machine learning text classifier is to transform the text extraction or text vectorization, and the classical approach has been bag-of-words or bag-of-ngrams with their frequency.

More recently, new feature extraction techniques have been applied based on word embeddings (also known as *word vectors*). This kind of representations makes it possible for words with similar meaning to have a similar representation, which can improve the performance of classifiers.

Classification Algorithms

The classification step usually involves a statistical model like Naïve Bayes, Logistic Regression, Support Vector Machines, or Neural Networks:

- Naïve Bayes: a family of probabilistic algorithms that uses Bayes's Theorem to predict the category of a text.
- Linear Regression: a very well-known algorithm in statistics used to predict some value (Y) given a set of features (X).
- Support Vector Machines: a non-probabilistic model which uses a representation of text examples as points in a multidimensional space. Examples of different categories (sentiments) are mapped to distinct regions within that space. Then, new texts are assigned a category based on similarities with existing texts and the regions they're mapped to.
- Deep Learning: a diverse set of algorithms that attempt to mimic the human brain, by employing artificial neural networks to process data.

Hybrid Approaches

Hybrid systems combine the desirable elements of rule-based and automatic techniques into one system. One huge benefit of these systems is that results are often more accurate.

Sentiment Analysis Challenges

Sentiment analysis is one of the hardest tasks in natural language processing because even humans struggle to analyze sentiments accurately.

Subjectivity and Tone

There are two types of text: subjective and objective. Objective texts do not contain explicit sentiments, whereas subjective texts do. Say, for example, you intend to analyze the sentiment of the following two texts:

The package is nice.

The package is red.

Most people would say that sentiment is positive for the first one and neutral for the second one, right? All *predicates* (adjectives, verbs, and some nouns) should not be treated the same with respect to how they create sentiment. In the examples above, *nice* is more *subjective* than *red*.

Context and Polarity

All utterances are uttered at some point in time, in some place, by and to some people, you get the point. All utterances are uttered in context. Analyzing sentiment without context gets pretty difficult. However, machines cannot learn about contexts if they are not mentioned explicitly. One of the problems that arise from context is changes in polarity. Look at the following responses to a survey:

Everything of it.

Absolutely nothing!

Imagine the responses above come from answers to the question *What did you like about the event?* The first response would be positive and the second one would be negative, right? Now, imagine the responses come from answers to the question *What did you DISlike about the event?* The negative in the question will make sentiment analysis change altogether.

A good deal of preprocessing or postprocessing will be needed if we are to take into account at least part of the context in which texts were produced. However, how to preprocess or postprocess data in order to capture the bits of context that will help analyze sentiment is not straightforward.

Irony and Sarcasm

When it comes to *irony* and *sarcasm*, people express their negative sentiments using positive words, which can be difficult for machines to detect without having a thorough understanding of the context of the situation in which a feeling was expressed.

For example, look at some possible answers to the question, *Did you enjoy your shopping experience with us?*

Yeah, sure. So smooth!

Not one, but many!

What sentiment would you assign to the responses above? The first response with an exclamation mark could be negative, right? The problem is there is no textual cue that will help a machine learn, or at least question that sentiment since *yeah* and *sure* often belong to positive or neutral texts.

How about the second response? In this context, sentiment is positive, but we're sure you can come up with many different contexts in which the same response can express negative sentiment.

Comparisons

How to treat comparisons in sentiment analysis is another challenge worth tackling. Look at the texts below:

This product is second to none.

This is better than older tools.

This is better than nothing.

The first comparison doesn’t need any contextual clues to be classified correctly. It’s clear that it’s positive.

The second and third texts are a little more difficult to classify, though. Would you classify them as *neutral*, *positive*, or even *negative*? Once again, context can make a difference. For example, if the *‘older tools’* in the second text were considered useless, then the second text is pretty similar to the third text.

Emojis

There are two types of emojis according to Guibon et al.. *Western emojis* (e.g. :D) are encoded in only one or two characters, whereas *Eastern emojis* (e.g. ¯ \ (ツ) / ¯) are a longer combination of characters of a vertical nature. Emojis play an important role in the sentiment of texts, particularly in tweets.

You’ll need to pay special attention to character-level, as well as word-level, when performing sentiment analysis on tweets. A lot of preprocessing might also be needed. For example, you might want to preprocess social media content and transform both Western and Eastern emojis into tokens and whitelist them (i.e. always take them as a feature for classification purposes) in order to help improve sentiment analysis performance.

Comprehensive list of emojis and their unicode characters that may come in handy when preprocessing.

Human Annotator Accuracy

Sentiment analysis is a tremendously difficult task even for humans. On average, inter-annotator agreement (a measure of how well two (or more) human labelers can make the same annotation decision).is pretty low when it comes to sentiment analysis. And since machines learn from the data they are fed, sentiment analysis classifiers might not be as precise as other types of classifiers.

Still, sentiment analysis is worth the effort, even if your sentiment analysis predictions are wrong from time to time.

Sentiment Analysis Applications

The applications of sentiment analysis are endless and can be applied to any industry, from finance and retail to hospitality and technology. The most popular applications of sentiment analysis in business:

1. Social Media Monitoring
2. Brand Monitoring
3. Voice of customer (VoC)
4. Customer Service
5. Market Research

Social Media Monitoring

Sentiment analysis is used in social media monitoring, allowing businesses to gain insights about how customers feel about certain topics, and detect urgent issues in real time before they spiral out of control.

On the fateful evening of April 9th, 2017, United Airlines forcibly removed a passenger from an overbooked flight. The nightmare-ish incident was filmed by other passengers on their smartphones and posted immediately. One of the videos, posted to Facebook, was shared more than 87,000 times and viewed 6.8 million times by 6pm on Monday, just 24 hours later.

The fiasco was only magnified by the company's dismissive response. On Monday afternoon, United's CEO tweeted a statement apologizing for "having to re-accommodate customers."

This is exactly the kind of PR catastrophe you can avoid with sentiment analysis. It's an example of why it's important to care, not only about if people are talking about your brand, but how they're talking about it. More mentions don't equal positive mentions.

Brands of all shapes and sizes have meaningful interactions with customers, leads, even their competition, all across social media. By monitoring these conversations you can understand customer sentiment in real time and over time, so you can detect disgruntled customers immediately and respond as soon as possible.

Brand Monitoring

Not only do brands have a wealth of information available on social media, but across the internet, on news sites, blogs, forums, product reviews, and more. Again, we can look at not just the volume of mentions, but the individual and overall quality of those mentions.

In our United Airlines example, for instance, the flare-up started on the social media accounts of just a few passengers. Within hours, it was picked up by news sites and spread like wildfire across the US, then to China and Vietnam, as United was accused of racial profiling against a passenger of Chinese-Vietnamese descent. In China, the incident became the number one trending topic on Weibo, a microblogging site with almost 500 million users.

And again, this is all happening within mere hours of the incident.

Brand monitoring offers a wealth of insights from conversations happening about your brand from all over the internet. Analyze news articles, blogs, forums, and more to guage brand sentiment, and target certain demographics or regions, as desired. Automatically categorize the urgency of all brand mentions and route them instantly to designated team members.

Get an understanding of customer feelings and opinions, beyond mere numbers and statistics. Understand how your brand image evolves over time, and compare it to that of your competition. You can tune into a specific point in time to follow product releases, marketing campaigns, IPO filings, etc., and compare them to past events.

Sentiment analysis allows you to automatically monitor all chatter around your brand and detect and address this type of potentially-explosive scenario while you still have time to defuse it.

Voice of Customer (VoC)

Social media and brand monitoring offer us immediate, unfiltered, and invaluable information on customer sentiment, but you can also put this analysis to work on surveys and customer support interactions.

Net Promoter Score (NPS) surveys are one of the most popular ways for businesses to gain feedback with the simple question: Would you recommend this company, product, and/or service to a friend or family member? These result in a single score on a number scale.

Businesses use these scores to identify customers as promoters, passives, or detractors. The goal is to identify overall customer experience, and find ways to elevate all customers to "promoter" level, where they, theoretically, will buy more, stay longer, and refer other customers.

Numerical (quantitative) survey data is easily aggregated and assessed. But the next question in NPS surveys, asking why survey participants left the score they did, seeks open-ended responses, or qualitative data.

Open-ended survey responses were previously much more difficult to analyze, but with sentiment analysis these texts can be classified into positive and negative (and everywhere in between) offering further insights into the Voice of Customer (VoC).

Sentiment analysis can be used on any kind of survey – quantitative and qualitative – and on customer support interactions, to understand the emotions and opinions of your customers. Tracking customer sentiment over time adds depth to help understand why NPS scores or sentiment toward individual aspects of your business may have changed.

You can use it on incoming surveys and support tickets to detect customers who are 'strongly negative' and target them immediately to improve their service. Zero in on certain demographics to understand what works best and how you can improve.

Real-time analysis allows you to see shifts in VoC right away and understand the nuances of the customer experience over time beyond statistics and percentages.

Discover how we analyzed the sentiment of thousands of Facebook reviews, and transformed them into actionable insights.

Customer Service

We all know the drill: stellar customer experiences means a higher rate of returning customers. Leading companies know that how they deliver is just as, if not more, important as what they deliver. Customers expect their experience with companies to be immediate, intuitive, personal, and hassle-free. If not, they'll leave and do business elsewhere. Did you know that one in three customers will leave a brand after just one bad experience?

You can use sentiment analysis and text classification to automatically organize incoming support queries by topic and urgency to route them to the correct department and make sure the most urgent are handled right away.

Analyze customer support interactions to ensure your employees are following appropriate protocol. Increase efficiency, so customers aren't left waiting for support. Decrease churn rates; after all it's less hassle to keep customers than acquire new ones.

Market Research

Sentiment analysis empowers all kinds of market research and competitive analysis. Whether you're exploring a new market, anticipating future trends, or seeking an edge on the competition, sentiment analysis can make all the difference.

You can analyze online reviews of your products and compare them to your competition. Maybe your competitor released a new product that landed as a flop. Find out what aspects of the product performed most negatively and use it to your advantage.

Follow your brand and your competition in real time on social media. Locate new markets where your brand is likely to succeed. Uncover trends just as they emerge, or follow long-term market leanings through analysis of formal market reports and business journals.

You'll tap into new sources of information and be able to quantify otherwise qualitative information. With social data analysis you can fill in gaps where public data is scarce, like emerging markets.

Data Cleaning:

The steps used to complete cleaning our data were:

1. Make text lowercase

   It is necessary to convert the text to lower case as it is case sensitive.

   text = "This is a Demo Text for NLP using NLTK. Full form of NLTK is Natural Language Toolkit"
   lower_text = text.lower()

2. **word tokenize**

Tokenize sentences to get the tokens of the text i.e breaking the sentences into words.

```
text = "This is a Demo Text for NLP using NLTK. Full form of NLTK is Natural Language Toolkit"
word_tokens = nltk.word_tokenize(text)
print (word_tokens)

[OUTPUT]: ['This', 'is', 'a', 'Demo', 'Text', 'for', 'NLP', 'using', 'NLTK', '.', 'Full', 'form', 'of', 'NLTK', 'is', 'Natural', 'Language', 'Toolkit'].
```

3. *sent tokenize*

Tokenize sentences if the there are more than 1 sentence i.e breaking the sentences to list of sentence.

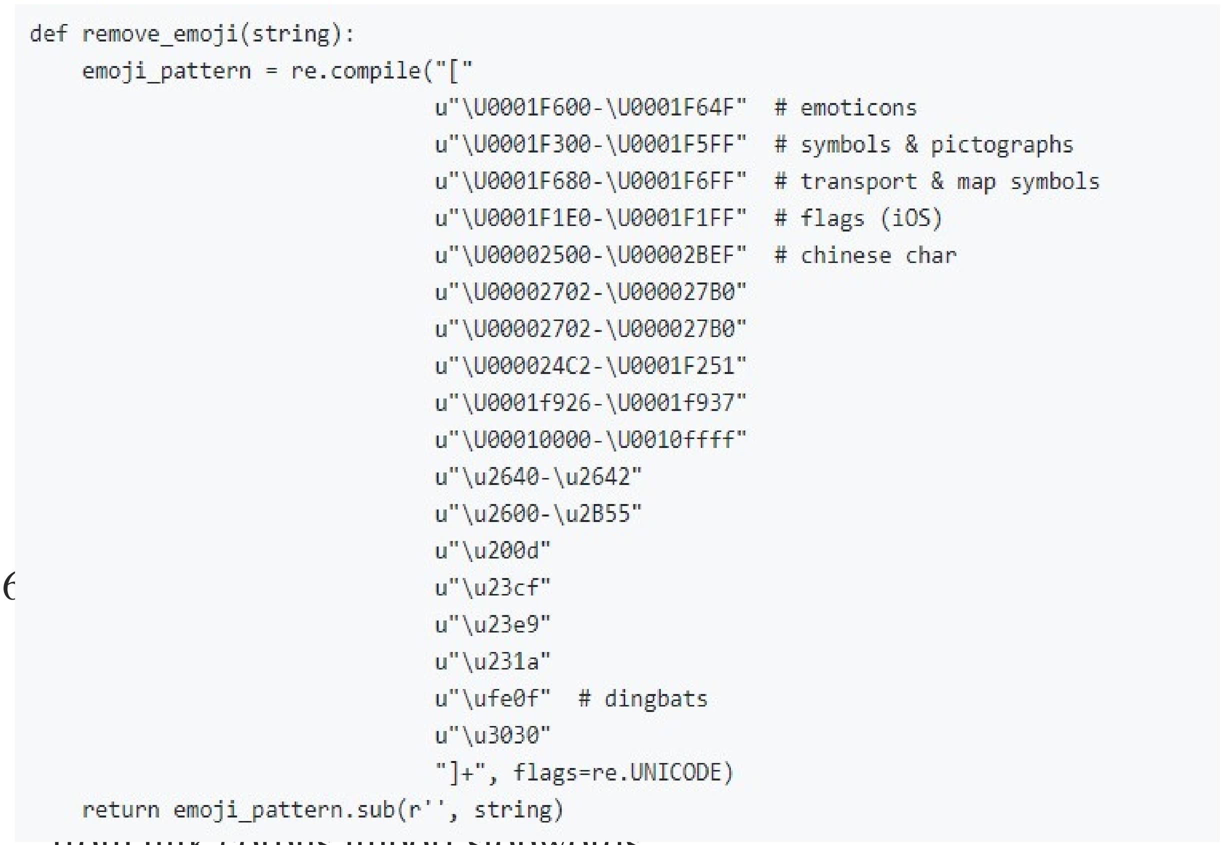
```
text = "This is a Demo Text for NLP using NLTK. Full form of NLTK is Natural Language Toolkit"
sent_token = nltk.sent_tokenize(text)
print (sent_token)
```

[OUTPUT]: ['This is a Demo Text for NLP using NLTK.', 'Full form of NLTK is Natural Language Toolkit']

4. Remove punctuation

```
sentence = "Think and wonder, wonder and think."
words = nltk.word_tokenize(sentence)
new_words= [word for word in words if word.isalnum()]
print(new_words)
```

5. Remove emoji's:



```
def remove_emoji(string):
    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002500-\U00002BEF"  # chinese char
                           u"\U00002702-\U000027B0"
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           u"\U0001f926-\U0001f937"
                           u"\U00010000-\U0010ffff"
                           u"\u2640-\u2642"
                           u"\u2600-\u2B55"
                           u"\u200d"
                           u"\u23cf"
                           u"\u23e9"
                           u"\u231a"
                           u"\ufe0f"  # dingbats
                           u"\u3030"
                           "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', string)
```

6

```
from nltk.corpus import stopwords
stopword = stopwords.words('english')
text = "This is a Demo Text for NLP using NLTK. Full form of NLTK is Natural Language Toolkit"
word_tokens = nltk.word_tokenize(text)
removing_stopwords = [word for word in word_tokens if word not in stopword]
```

```
print (removing_stopwords)
[OUTPUT]: ['This', 'Demo', 'Text', 'NLP', 'using', 'NLTK', '.', 'Full', 'form', 'NLTK', 'Natural', 'Language', 'Toolkit']
```

7. Lemmatization

   lemmatize the text so as to get its root form eg: functions,funtionality as function

```
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
#is based on The Porter Stemming Algorithm
stopword = stopwords.words(‘english’)
wordnet_lemmatizer = WordNetLemmatizer()
text = “the dogs are barking outside. Are the cats in the garden?”
word_tokens = nltk.word_tokenize(text)
lemmatized_word = [wordnet_lemmatizer.lemmatize(word) for word in word_tokens]
print (lemmatized_word)
[OUTPUT]: ['the', 'dog', 'are', 'barking', 'outside', '.', 'Are', 'the', 'cat', 'in', 'the', 'garden', '?']
```

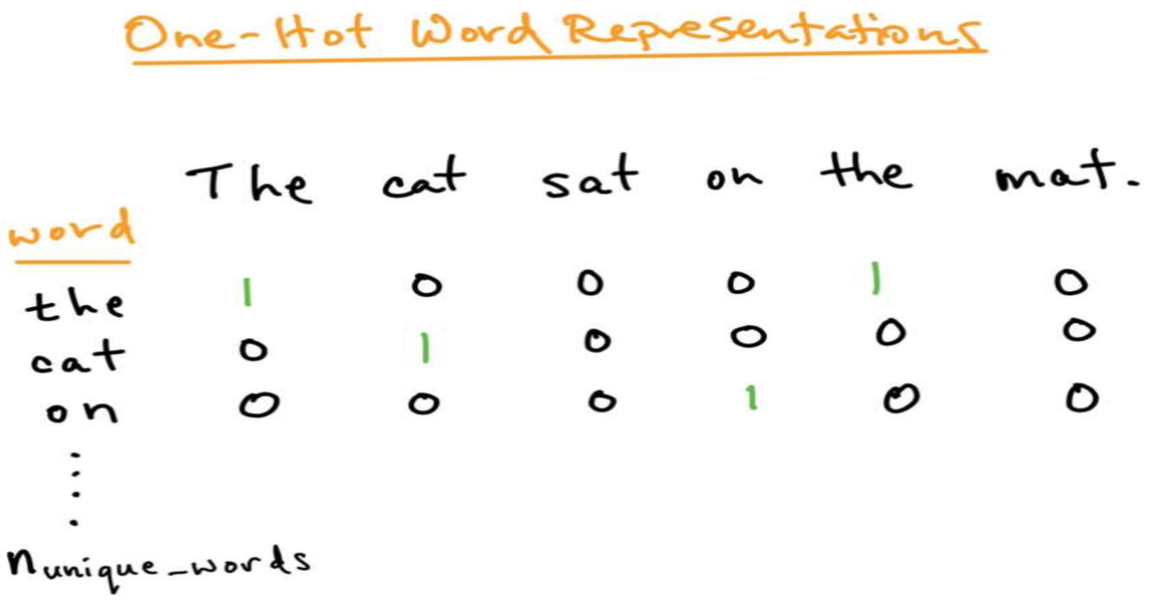
Text Representation:

It is important to transform the **text corpus** into different input formats for a **Machine Learning model**. Starting from the left, the Corpus goes through several steps before obtaining the **Tokens**, a set of text **building blocks** i.e. **words**, **subwords**, **characters**, etc. Since ML models are only capable of processing **numerical values.**

1. One Hot Encoding:

One-hot encoding may be one’s go-to move when a **categorical feature** is encountered in a data set. It’s a simple and straight forward technique and it works by replacing each category with a **vector** full of **zeros**, except for the position of its corresponding **index** value, which has a value of **1**.

When applying one-hot encoding to a text document, tokens are replaced by their one-hot vectors and a given sentence is in turn transformed into a **2D-matrix** with a shape of ***(n, m)***, with ***n*** being the **number of token** in the sentence and ***m*** the **size** of the **vocabulary**. Depending on the number of tokens a sentence has, its **shape** will be **different**.



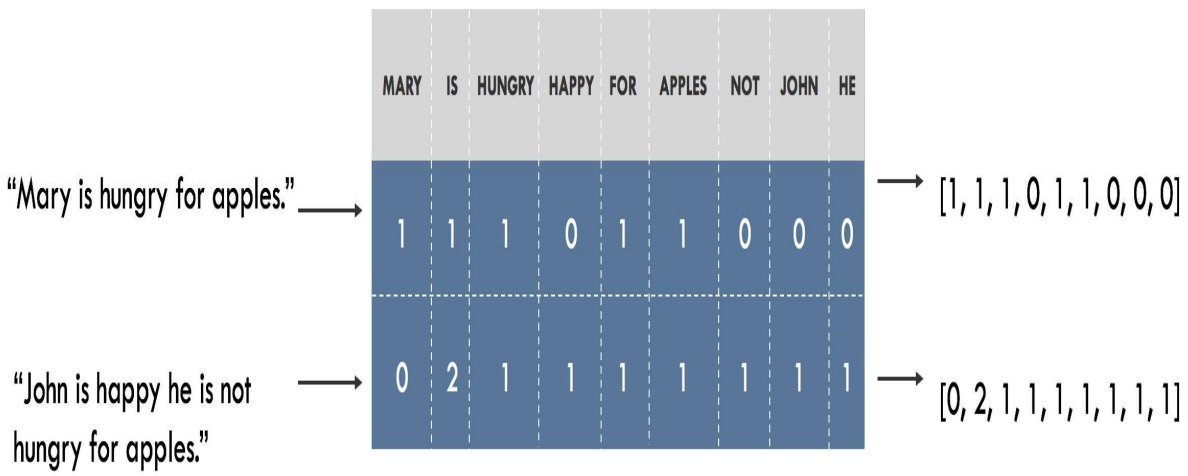
One-Hot Word Representations

word	The	cat	sat	on	the	mat.
the	1	0	0	0	1	0
cat	0	1	0	0	0	0
on	0	0	0	1	0	0
⋮						
n_{unique_words}						

2. Bag of Words Representation:

Bag-of-words representation as the name suggests intutively, puts words in a “bag” & computes frequency of occurrence of each word. It does not take into account the word order or lexical information for text representation

The intuition behind BOW representation is that document having similar words are similar irrespective of the word positioning

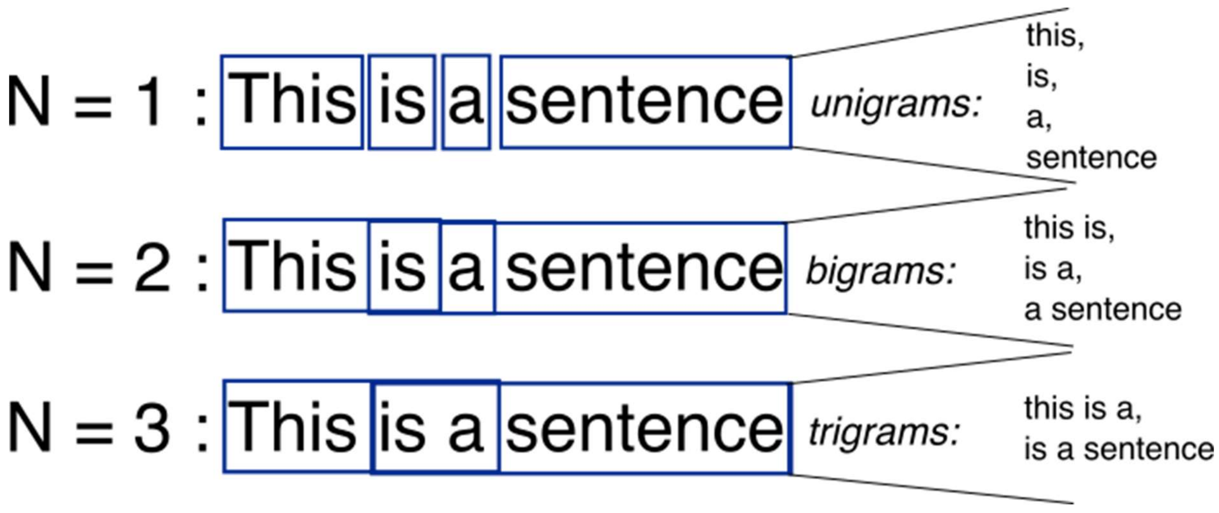


	MARY	IS	HUNGRY	HAPPY	FOR	APPLES	NOT	JOHN	HE	
"Mary is hungry for apples."	1	1	1	0	1	1	0	0	0	[1, 1, 1, 0, 1, 1, 0, 0, 0]
"John is happy he is not hungry for apples."	0	2	1	1	1	1	1	1	1	[0, 2, 1, 1, 1, 1, 1, 1, 1]

3. N-grams

N-gram is probably the easiest concept to understand in the whole machine learning space, I guess. An N-gram means a sequence of N words. So for example, "Medium blog" is a 2-gram (a bigram)

Before we move on to the probability stuff, let's answer this question first. Why is it that we need to learn n-gram and the related probability? Well, in Natural Language Processing, or NLP for short, n-grams are used for a variety of things. Some examples include auto completion of sentences (such as the one we see in Gmail these days), auto spell check (yes, we can do that as well), and to a certain extent, we can check for grammar in a given sentence.



4. WORD EMBEDDING
   Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words.

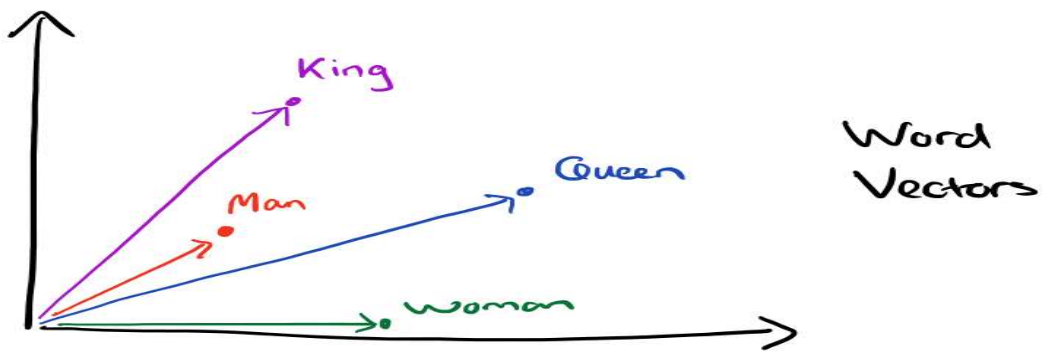
Consider the following similar sentences: *Have a good day* and *Have a great day*. They hardly have different meaning. If we construct an exhaustive vocabulary (let's call it V), it would have V = {Have, a, good, great, day}.

Now, let us create a one-hot encoded vector for each of these words in V. Length of our one-hot encoded vector would be equal to the size of V (=5). We would have a vector of zeros except for the element at the index representing the corresponding word in the vocabulary. That particular element would be one. The encodings below would explain this better.

Have = [1,0,0,0,0]`; a=[0,1,0,0,0]` ; good=[0,0,1,0,0]` ; great=[0,0,0,1,0]` ; day=[0,0,0,0,1]` (` represents transpose)

If we try to visualize these encodings, we can think of a 5 dimensional space, where each word occupies one of the dimensions and has nothing to do with the rest. This means ‘good’ and ‘great’ are as different as ‘day’ and ‘have’, which is not true.

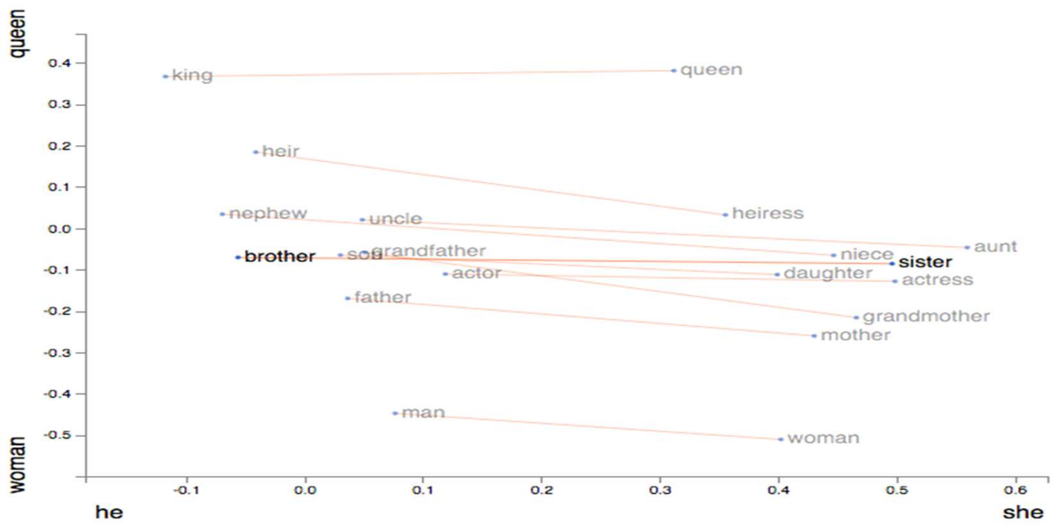
Our objective is to have words with similar context occupy close spatial positions. Mathematically, the cosine of the angle between such vectors should be close to 1, i.e. angle close to 0.



Word embedding is a mapping of words into vectors

- A word gets converted into a vector of N dimensions, where each of these dimensions will have an underlying meaning. How the word relates to these underlying meanings will define the values for the dimension.
- A word embedding of 2 dimensions may map words as follows:

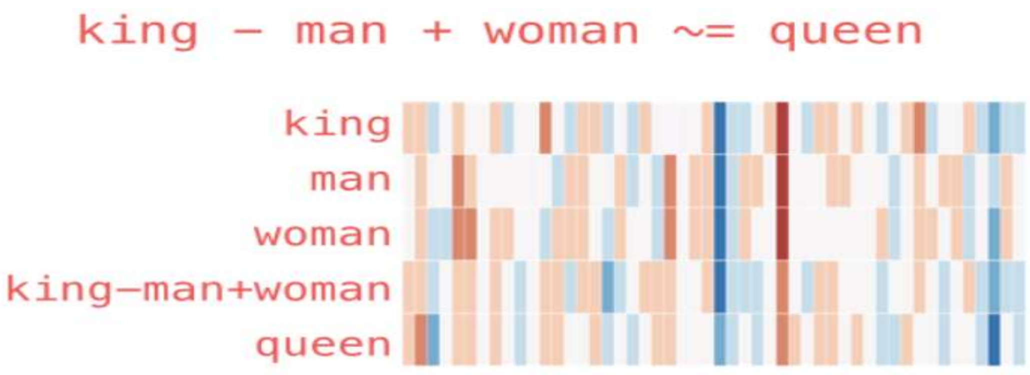
    man → [0.85, 0.05]
    dog → [0.9, 0.1]
    apple → [0.12, 0.9]
    banana → [0.06, 0.95]



If we want to teach it to a computer, the simplest, approximated approach is making it look only at word pairs. Let *P(a|b)* be the conditional probability that given a word *b* there is a word *a* within a short distance. Then we claim that two words *a* and *b* are similar if

P(w|a)=P(w|b)

for every word *w*. In other words, if we have this equality, no matter if there is word *a* or *b*, all other words occur with the same frequency.


king − man + woman ~= queen
king
man
woman
king−man+woman
queen