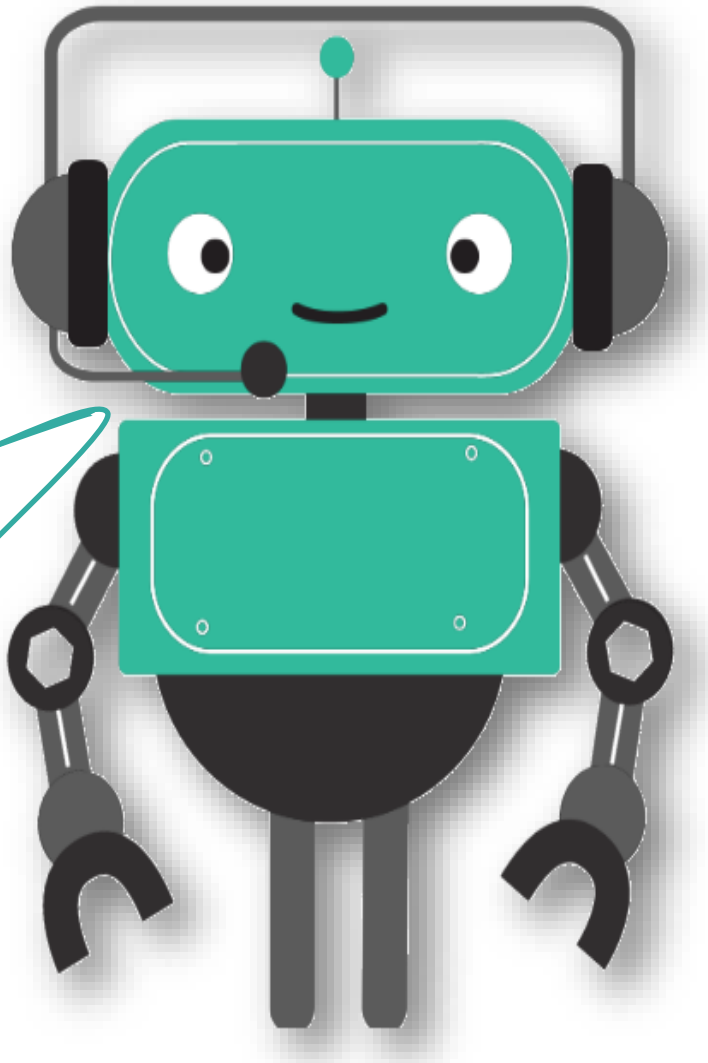
Analytics Vidhya




# Natural Language Processing

- Analytics Vidhya

# Introduction

The amount of data we produce today is astounding! With so much information at our fingertips, we are creating data every time we seek answers from the internet.

> **"According to Forbes, There are 2.5 quintillion bytes of data created each day at our current pace"**

Our friends in this mission to multiply the creation of data are gadgets like Smartphones, Tabs, etc. They are propelling this pace in a mind-boggling manner. Every interaction, WhatsApp message, comments, reactions on social media posts are recorded and that's a humongous chunk of data. Some of the most common examples of such data are – tweets/posts on social media, the user to user chat conversations, news feed, and reviews and patient records in the healthcare sector. A few more recent examples of where data is continuously being generated include chatbots and other voice-driven bots such as Alexa and Siri.

This e-book will help in-

- In-depth understanding of the world of Natural Language Processing (NLP).
- Building your own Natural Language Processing Model
- Wish to build a Career in NLP? The answer is here

# Table of Contents

# What is Natural Language Processing?

Natural Language Processing is a branch of data science that consists of systematic processes for analyzing, understanding, and deriving information from the text data. By using the techniques of Natural Language Processing, one can organize and analyze the massive chunks of text data, perform numerous automated tasks to solve a wide range of problems such as – automatic summarization, machine translation, and many more.

# Applications of Natural Language Processing in our day-to-day life

Natural Language Processing is essentially teaching machines to understand human language and since it is all about human language, you come across multiple applications of NLP in your daily life without even realizing it! Here are a few examples that you would have definitely come across:

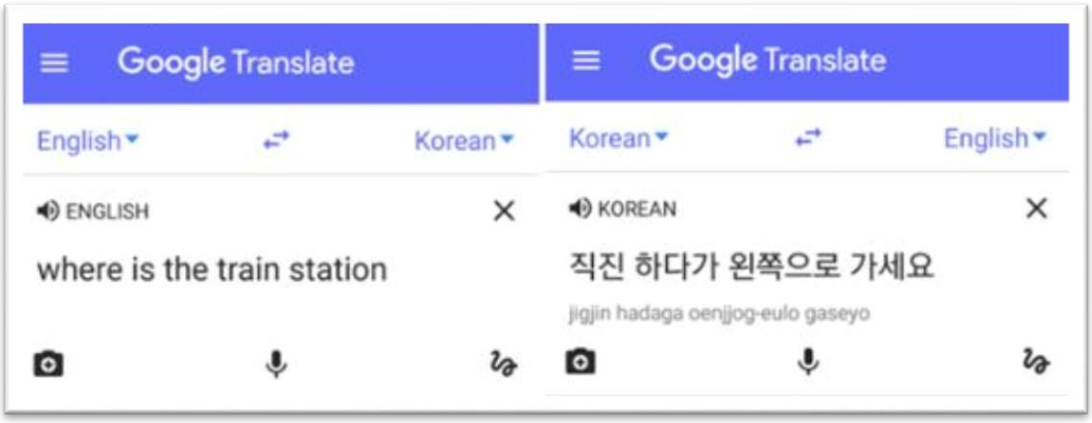
## 1. Chatbots or Conversational Agents

Chatbots are everywhere today, from booking your flight tickets to ordering food, chances are that you have already interacted with one.


Customers nowadays don't want to wait for hours just to get their queries resolved. They want instant answers and chatbots come in really handy here, both for your business as well as your customers.
Similarly, we have many conversational agents or AI assistants like Alexa, Siri, Cortana and Google Home that uses natural language processing internally.

## 2. Machine Translation

A machine translation system uses natural language processing techniques in collaboration with Machine learning/Deep-Learning to build systems that are capable of automatic language translation.

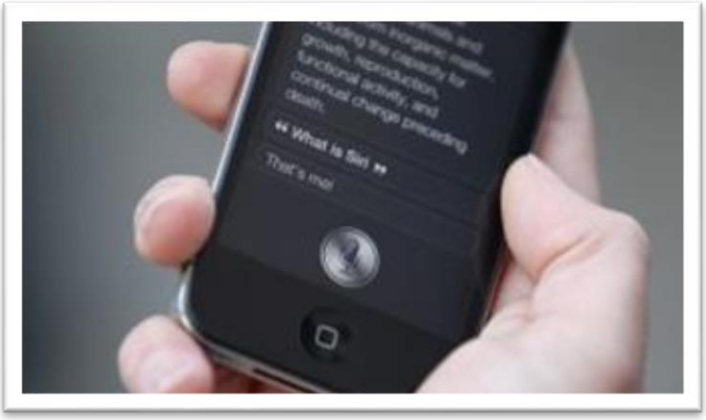



If you have ever read a post in another language on Facebook and seen its translation just below it, or opened a website of any language other than English in Chrome or even used Google Translate on a trip to a foreign country then you have used some kind of a machine translation system. Applications like Google Translate are a very good example of what we call as Neural Machine Translation (NMT), these are language translation systems that are built on top of Neural Networks.

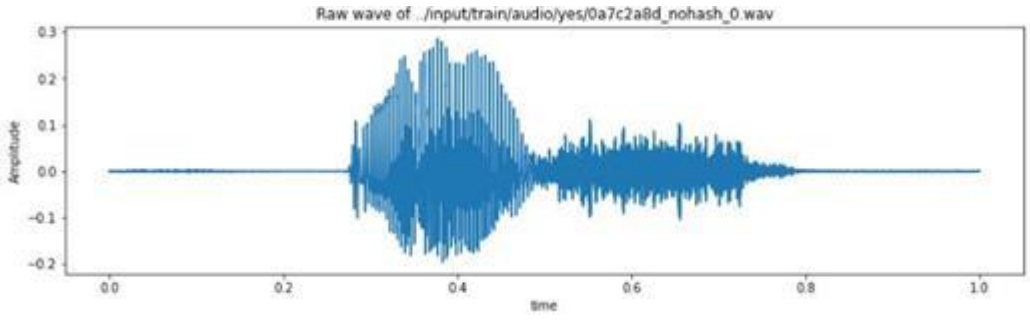

### 3. Speech Recognition

Voice-based personal assistants have become so ubiquitous in the past decade that almost every Smartphone user would be very familiar with the likes of Apple's Siri, Amazon's Alexa, Baidu's Deur, and Google's Assistant. But have you ever wondered about the technology that powers

these assistants under the hood? Siri and Google can easily understand what you are saying, how does the system convert your query into text on your phone's screen?



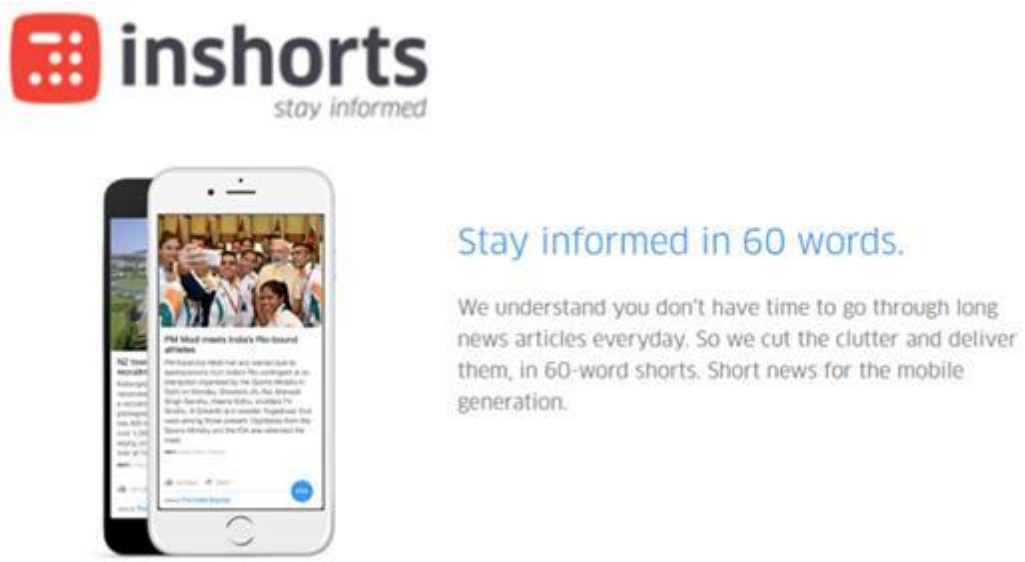
This is exactly where Natural Language Processing comes in the picture, every audio that you speak can be considered as "natural language" and if you can convert this audio to text then all you have to do is reuse the models and techniques you used for building the text-based NLP systems. For example, if you want to build a voice-based AI assistant, then all you have to do is convert the audio interaction (commands) into text and feed it into the text-based Chatbot that you have. From there onwards, your Chatbot can take over the interaction with the person and you just have to take care of the part where you convert from text to speech and vice-versa. Similarly, speech recognition becomes a simple text classification once you can convert the audio to text. The same is true for language translation.




## 4. Text Summarization

In today's busy world, people need byte sized summaries of information to effectively take action on it without indulging time more than necessary. Text summarization is one such application of Natural

Language Processing (NLP) that is slowly becoming the need of the hour. Who has the time to go through a myriad of articles/documents /bookstores decide whether they are useful or not? Wouldn't it help if instead of reading long medical or legal reports, you can simply get the summary of the proceedings with important points? Thankfully – this technology is already here.




Automatic Text Summarization is one of the most interesting problems in the field of Natural Language Processing (NLP). It is a process of generating a concise, coherent and meaningful summary of text from text resources such as books, news articles, blog posts, research papers, etc. Automatic text summarization can broadly be divided into two categories — Extractive Summarization and Abstractive Summarization.

**a) Extractive Summarization:**

In this approach, several parts from the original text document are extracted, such as phrases and sentences, from a piece of text and stacked together to create a summary. Therefore, identifying the right sentences for summarization is of utmost importance in an extractive method.

### b) Abstractive Summarization

These methods use advanced NLP techniques to generate an entirely new summary. Some parts of this summary may not even appear in the original text.

## 5. Recommendation Engine

Recommendation Engines are everywhere, from online shopping giants like Amazon to online video streaming services like Netflix, everyone is trying to provide better recommendations to the user to improve their experience and at the same time to drive sales. A recommendation engine tries to understand a user's needs and interests using the data of the past behavior of the user and filters it using different algorithms to recommend the most relevant item/choice to users.


Interestingly now, you can use Natural Language Processing to create recommendation engines too. Take an example, suppose there is a customer who is buying sports-related items like a sweat-free T-shirt, shorts and tennis shoes. Now what do you think you should recommend the customer to buy next?

You can recommend similar sports-related accessories like a cap, water sipper and so on. Let's convert this problem into a text-based problem: if you want to recommend the next items the customer should buy you can just create a sentence made up of words of all the products that the customer has already bought and your task is to simply predict the next product he should buy or, the next word that should come in the sentence. This problem is now a sentence completion or next word prediction task.

Now we are not digging deep on how this system is built but if you want to learn how to create a recommendation engine using NLP, you can check the following article: Building a Recommendation System using Word2vec: A Unique Tutorial with Case Study in Python

# Applications of Natural Language Processing in the Industry

Now that you are already familiar with NLP based applications that you use in your daily life as a layman, let's understand what kind of applications of NLP will create value for businesses!

1. Sentiment analysis for customer reviews

Natural Language Processing (NLP) is a hotbed of research in data science these days and one of the most common applications of NLP is sentiment analysis. From opinion polls to creating entire marketing strategies, this domain has completely reshaped the way businesses work, which is why this is an area every data scientist must be familiar with. Thousands of text documents can be processed for the sentiment (and other features including named entities, topics, themes, etc.) in seconds, compared to the hours it would take a team of people to manually complete the same task.

If you want to learn how, can you mine user reviews or analyze sentiment from people's tweets, you can follow the below two articles:

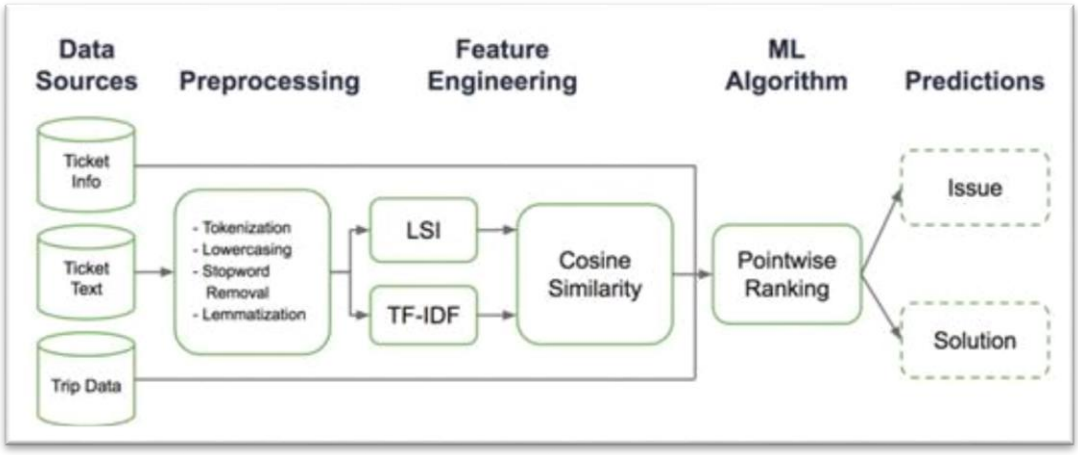
- An NLP Approach to Mining Online Reviews using Topic Modeling (with Python codes)
- Comprehensive Hands on Guide to Twitter Sentiment Analysis with dataset and code

## 2. Customer support systems

Companies like Uber that are large-scale and customer-facing have developed sophisticated systems for customer support requests. With hundreds of thousands of tickets surfacing daily on the platform across 400+ cities worldwide, their team must ensure that agents are empowered to resolve them as accurately and quickly as possible.

Enter COTA, Uber's Customer Obsession Ticket Assistant, a tool that uses machine learning and natural language processing (NLP) techniques to help agents deliver better customer support.

COTA enables quick and efficient issue resolution for more than 90% of Uber's inbound support tickets. This is because you are not only responding to a customer in real-time (now the first response is from a bot rather than a human agent) but also you are saving both the customer's and the agent's time by giving useful suggestions to the agent regarding the kind of problem the customer could be facing and the possible steps that could solve that problem. This is a very interesting use-case of natural language processing in the real world.

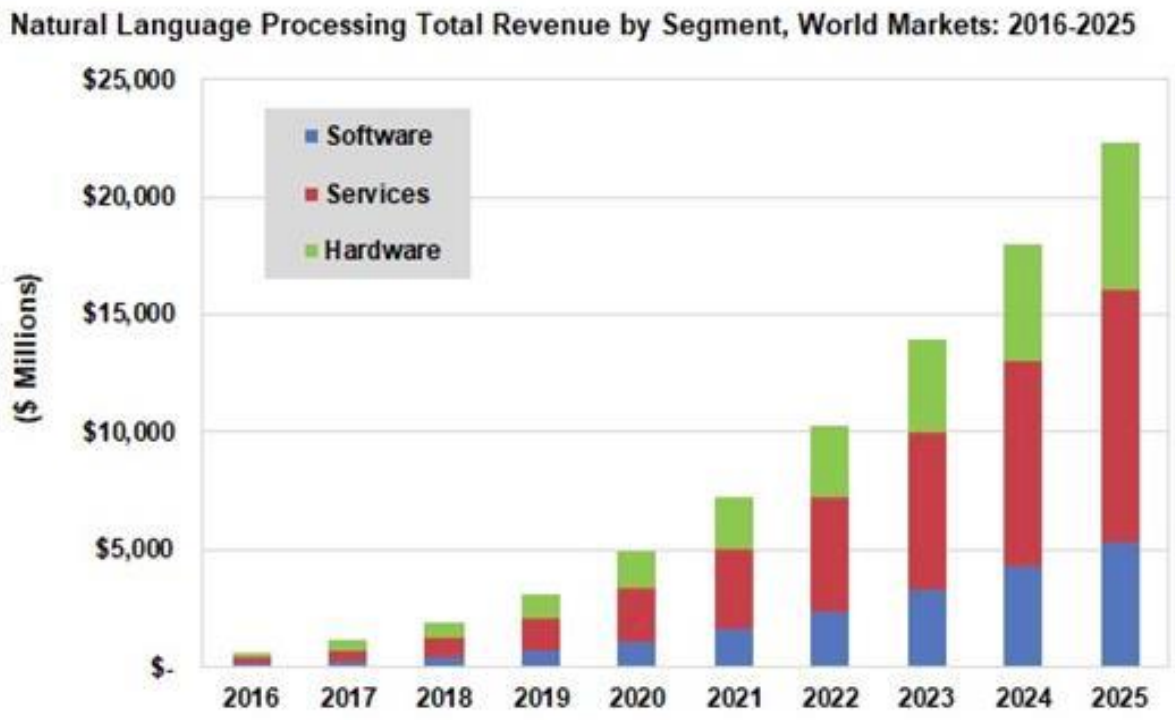


## 3. Text analytics

One of the biggest breakthroughs required for achieving any level of artificial intelligence is to have machines that can process text data. Thankfully, the amount of text data being generated in this universe has exploded exponentially in the last few years

It has become imperative for an organization to have a structure in place to mine actionable insights from the text being generated. From social media analytics to risk management and cybercrime protection, dealing with text data has never been more important.

# Why is Natural Language Processing getting so much Attention lately?




The above chart predicts the size of Natural Language Processing's market up to 2025 and while it is obvious that the market will continue to grow, the $ figures are extraordinary. The meteoric rise of NLP in today's time can be attributed to three major reasons:

1) Growth in unstructured data:

Since we are generating tons of unstructured data every second in the form of social media conversations, digital reports etc. there is a real need of developing the means to work with this data and make sense of it and that necessity is giving the necessary push to Natural Language Processing.

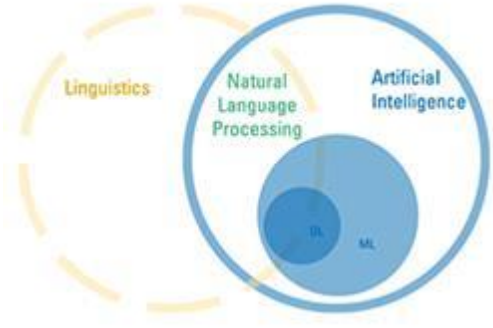
## 2) Rise of Deep Learning

These are times of Deep Learning which has shown to perform well in learning tasks, this rise of interest in DL research combined with the latest advancements has been a boon for NLP. Currently, Deep Learning is used actively across all the major domains and subdomains of NLP.

## 3) Evolution of User Interfaces

With improvement in technology we are experiencing better user interface and that has directly increased the motivation for NLP research because Natural language is the best interface for a user.

# How is Natural Language Processing different from Machine Learning, Deep Learning and Artificial Intelligence?




The diagram is an accurate representation of the relationship between Natural Language Processing, Deep Learning, Machine Learning and Artificial Intelligence.

## Artificial Intelligence

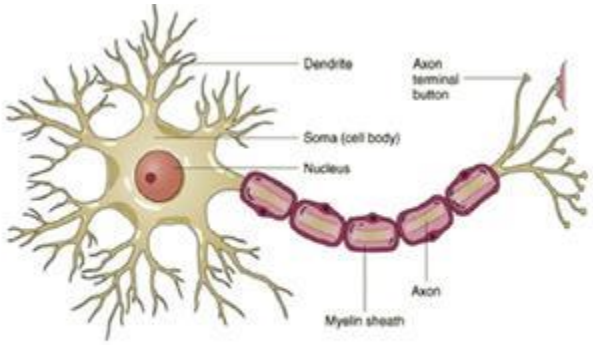
Artificial Intelligence is the umbrella term for all technologies that enable a machine to "think", as the name suggests, it can be loosely interpreted to mean incorporating human intelligence into machines. Intelligent behavior is when a machine can do tasks in a way that a human would do, on its own by following a similar reasoning pattern and way of thinking.

## Machine Learning

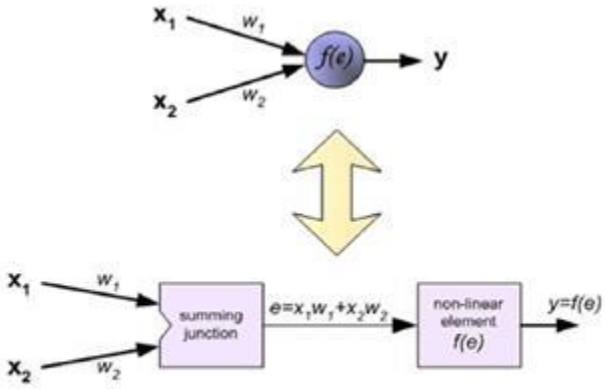
Machine Learning involves empowering computer systems with the ability to “learn". ML intends to enable machines to learn by themselves using the provided data and made accurate predictions. ML is a subset of artificial intelligence, it's simply a technique for realizing AI. It is a method of training algorithms such that they can learn how to make decisions.

## Deep learning

Deep Learning is a subset of ML, it's a technique for realizing machine learning. DL is considered the next evolution of machine learning. DL algorithms are roughly inspired by the information processing patterns found in the human brain. This is done based on individual units of intelligence that are known as Neural Networks.




(a) Biological neuron




(b) Artificial neural network

## Natural Language

Now you must be wondering how all of them are related to Natural Language Processing? The simple answer is that NLP uses a combination of these approaches. NLP is a direct cross over between Computational Linguistics and Artificial Intelligence and so uses the concepts and techniques that are developed in both the fields to solve inter-disciplinary problems.

# What tools/libraries are used in Natural Language Processing?

1. **Regular Expressions (REGEX)**
   A regular expression or regex is a sequence of characters that define a search pattern - WikipediaRegular Expressions use patterns to extract information from a given piece of text. At the same time, they are used for other useful NLP tasks like cleaning/filtering unnecessary symbols and searching for a given pattern in the text. You can learn more about how REGEX is used here:
   Extracting information from reports using Regular Expressions Library in Python

2. **NLTK**
   New Language Tool Kit or NLTK is one of the most popular NLP libraries in Python. It supports a plethora of tasks and can be used to do anything from text preprocessing techniques like stop words removal, tokenization, stemming, lemmatization to building n-grams. You can learn more about how to use NLTK for NLP in this article:
   Ultimate Guide to Understand and Implement Natural Language Processing (with codes in Python)

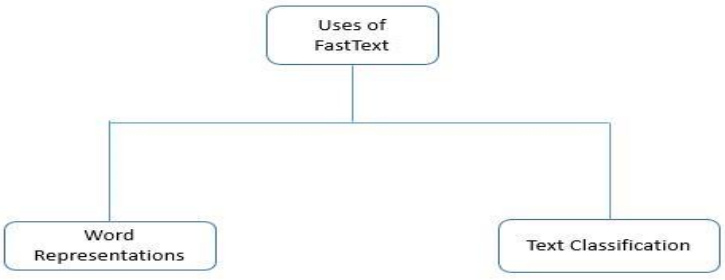
3. **spaCy**
   spaCy is considered as a successor to NLTK and is known as an industrial grade natural language processing library. It is scalable and uses the latest neural network based models to perform tasks like named entity recognition, parts of speech tagging, sentence dependency mapping, etc. Here is a nice article on how can you use spaCy with Python to do some useful NLP tasks:
   Natural Language Processing Made Easy – using spaCy (in Python)

## 4. Gensim

Gensim is an open-source library for unsupervised topic modeling and natural language processing, using modern statistical machine learning.
It is extensively used when working with word embeddings like Word2V1ec and Doc2Vec and also when one has to perform topic modeling related tasks. You can find more info about Gensim here:
Genism Tutorials

## 5. Fast Text

FastText is a library created by the Facebook Research Team for efficient learning of word representations and sentence classification. This library has gained a lot of traction in the NLP community and is a possible substitution to the gensim package which provides the functionality of Word Vectors etc. You can know more about FastText and how it can be effectively used for NLP here:
Text Classification & Word Representations using FastText (An NLP library by Facebook)



## 6. TextBlobs

TextBlobs is a beginner-friendly NLP library that is built on the shoulders of NLTK and Pattern. A big advantage of this is, it is easy to learn and offers a lot of features like sentiment analysis, POS- tagging, noun phrase extraction, etc. If it is your first step in NLP, TextBlob is the perfect library for you to get hands-on with. You can get a good start on TextBlob with this article: Natural Language Processing for Beginners: Using TextBlob

## 7. Stanford NLP

Stanford NLP is a library that is straight out of Stanford's NLP Research Group and the most interesting fact about this library is that it lets you perform text preprocessing on more than 53 human languages! Adding to that, it is incredibly fast and serves as an interface to the legendary NLP toolkit from Stanford that is Core NLP tools. Check out Stanford NLP here: Introduction to StanfordNLP: An Incredible State-of-the-Art NLP Library for 53 Languages (with Python code)


## 8. Flair

Flair is a simple natural language processing (NLP) library developed and open-sourced by Zalando Research. Flair's framework builds directly on PyTorch. The Zalando Research team has also released several pre-trained models for the following NLP tasks:

- Name-Entity Recognition (NER): It can recognize whether a word represents a person, location or names in the text.
- Parts-of-Speech Tagging (PoS): Tags all the words in the given text as to which "part of speech" they belong to.

- Text Classification: Classifying text based on the criteria (labels)
- Training Custom Models: Making our custom models.

Learn more about how to use Flair for NLP here: Introduction to Flair for NLP: A Simple yet Powerful State-of-the-Art NLP Library

9. Flash Text

Regex can sometimes be really slow when working on large documents, here is a new library that is faster than regular expressions for NLP preprocessing tasks: FlashText! FlashText is a Python library created specifically for the purpose of searching and replacing words in a document. Now, the way FlashText works is that it requires a word or a list of words and a string. The words which FlashText calls keywords are then searched or replaced in the string.

Here is a good tutorial on FlashText: FlashText – A library faster than Regular Expressions for NLP tasks

10. Transformers by HuggingFace

This library is good for people who want to try the latest groundbreaking models in NLP without waiting for it. Released this year only, Pytorch-Transformers brings state-of-the-art NLP models like BERT, XLNet and Transformers-XL to Python.
Learn how to work with this amazing library here:
Introduction to PyTorch-Transformers: An Incredible Library for State-of-the-Art NLP (with Python code)

11. Other widely used Libraries:

Here are some libraries that aren't as widely used as the ones we discussed but deserve recommendation:

- polyglot
- pywsd
- TextBlob
- pattern
- vocabulary
- pynlpi
- query
- CoreNLP

# Different Techniques used in Natural Language Processing

## 1. Part of Speech Tagging or POS

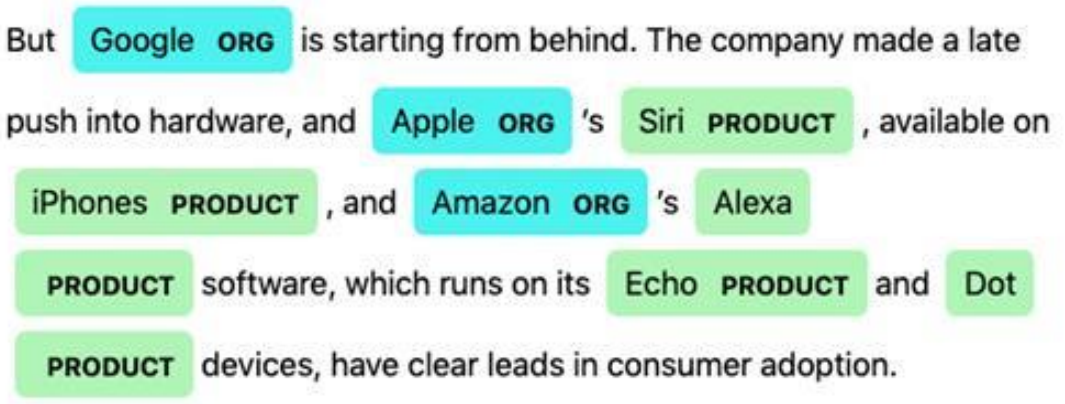
Part of Speech Tagging or POS Tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context—i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph (nouns, verbs, adjectives, adverbs, etc)

| TEXT | LEMMA | POS | TAG | DEP | SHAPE | ALPHA | STOP |
|---|---|---|---|---|---|---|---|
| Apple | apple | PROPN | NNP | nsubj | Xxxxx | True | False |
| is | be | VERB | VBZ | aux | xx | True | True |
| looking | look | VERB | VBG | ROOT | xxxx | True | False |
| at | at | ADP | IN | prep | xx | True | True |
| buying | buy | VERB | VBG | pcomp | xxxx | True | False |
| U.K. | u.k. | PROPN | NNP | compound | X.X. | False | False |
| startup | startup | NOUN | NN | dobj | xxxx | True | False |
| for | for | ADP | IN | prep | xxx | True | True |
| $ | $ | SYM | $ | quantmod | $ | False | False |
| 1 | 1 | NUM | CD | compound | d | False | False |
| billion | billion | NUM | CD | probj | xxxx | True | False |

## 2. Name Entity Recognition

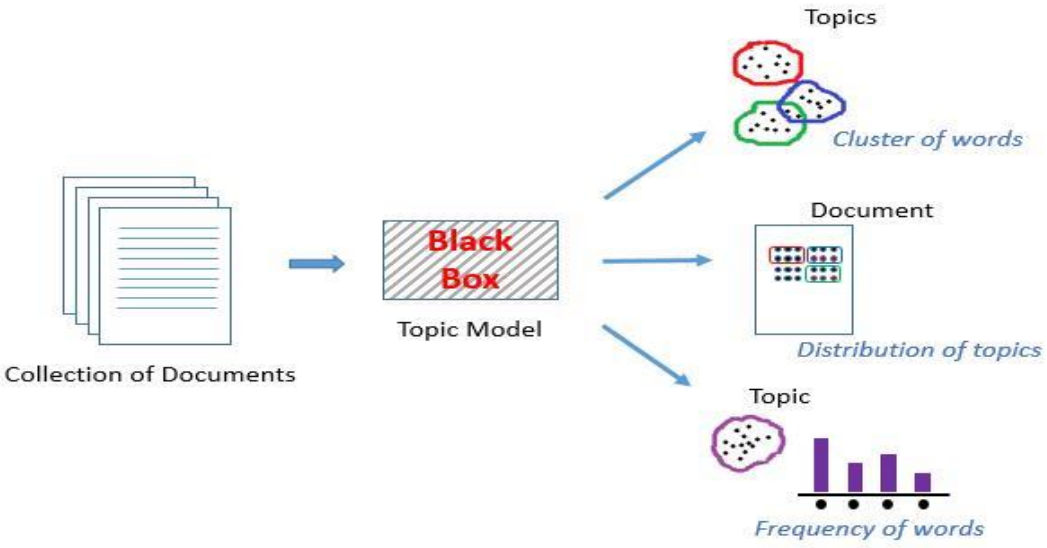
Named Entity Recognition or NER is the process of detecting the real-world named entities such as person names, location names, company names, etc from a (NER) given piece of text. For example: Sentence – Sergey Brin, the manager of Google Inc. is walking in the streets of New

York. Named Entities – ( “person” : “Sergey Brin” ), (“org” : “Google Inc.”), (“location” : “New York”)




## 3. Topic Modeling

Topic modelling is the process of automatically identifying the topics present in a text corpus, it derives the hidden patterns among the words in the corpus in an unsupervised manner. This can be useful for a lot of real-world applications, say you want to arrange millions of patient records by certain symptoms they face or the kind of treatment they are receiving then topic modeling can automate this task for you. Topics are defined as “a repeating pattern of co-occurring terms in a corpus”. A good topic model would result in terms like “health”, “doctor”, “patient”, “hospital” for a topic like Healthcare, and terms like “farm”, “crops”, “wheat” for another topic like agriculture. Now there are multiple algorithms and approaches to implement Topic Modeling, here are a few important ones:



## 4. Latent Dirichlet Allocation(LDA)

LDA is one of the most popular algorithms to implement Topic Modeling. LDA assumes documents are produced from a mixture of topics. Those topics then generate words based on their probability distribution. Given a dataset of documents, LDA backtracks and tries to figure out what topics would create those documents in the first place. You can learn more about LDA here: Beginners Guide to Topic Modeling in Python

## 5. Latent Semantic Analysis(LSA)

LSA assumes that words that are close in meaning will occur in similar pieces of text. A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns. Paragraphs are then compared by taking the cosine of the angle between the two vectors (or the dot product between the normalizations of the two vectors) formed by any two columns. Values close to 1 represent very similar paragraphs while values close to 0 represent very dissimilar paragraphs.

You can learn more about LSA here: Text Mining 101: A Stepwise Introduction to Topic Modeling using Latent Semantic Analysis (using Python)

## 6. Language Modeling

Language Modeling is the first crucial step for most of the advanced NLP tasks like Text summarization, Machine translation, Chatbots etc. It involves learning to predict the probability of a sequence of words. Now, this is the same technique that Google uses when it gives you search suggestions.

You can learn more about Language Modeling here: A Comprehensive Guide to build your own Language Model in Python!


5-gram


## 7. Sequence Modeling

Sequence Modeling is a technique of deep learning that is used to work with sequence data like music lyrics, sentence translation, understanding reviews or building chatbots. This technique is used a lot in NLP because natural language or text is essentially an example of sequence-based data. You can learn more about Sequence Modeling here: [Must-Read Tutorial to Learn Sequence Modeling (deeplearning.ai Course #5)]

# How much Data is required to train a model for Applying Natural Language Processing?

There is no single answer to this question. It depends on the problem you are trying to solve as different kinds of NLP problems require different amounts of data to work with. But here are some guidelines:

- In general – the golden rule is to collect as much data as possible. If the cost of collecting the data is not very high – this ends up working fine
- If the cost of capturing the data is high, then you would need to do a cost-benefit analysis based on the expected benefits coming from machine learning models
- The data required for basic NLP tasks where you are utilizing classical machine learning algorithms like SVM, Naive Bayes etc. would usually be lesser when compared to the amount of data required for complex NLP tasks like Text summarization or Chatbots.

# Where can you find Datasets/Pre-trained models for training Natural Language Processing systems?

Natural Language Processing is an exploding field so there is a lot of interest in gathering and collecting datasets and releasing pre-trained models. Here are some useful examples of the same:

**Datasets**

## 1. The General Language Understanding Evaluation


The General Language Understanding Evaluation (GLUE) benchmark is a collection of resources for training, evaluating, and analyzing natural language understanding systems. This dataset is so comprehensive that almost all the State-of-the-Art NLP systems first try their performance on this dataset and only after they have aced most or all of the tasks of GLUE that their models are considered as a new breakthrough. Here are some features of GLUE:

- A benchmark of nine sentence- or sentence-pair language understanding tasks built on established existing datasets and selected to cover a diverse range of dataset sizes, text genres, and degrees of difficulty,
- A diagnostic dataset designed to evaluate and analyze model performance with respect to a wide range of linguistic phenomena found in natural language, and
- A public leaderboard for tracking performance on the benchmark and a dashboard for visualizing the performance of models on the diagnostic set.

The GLUE benchmark's format is independent of the type of model you are using, so any system capable of processing sentence and sentence pairs and producing corresponding predictions is eligible to participate.

## 2. The Stanford Question Answering Dataset (SQUAD)


The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowd workers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable. This dataset is again a widely preferred dataset for testing State-of-the-Art NLP models.

Pre-trained models:
You can use the PyTorch-Transformers library to access most of the Pre-trained models in NLP. Here are some of the models it provides:

- BERT (from Google) released with the paperBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- GPT (from OpenAI) released with the paper Improving Language Understanding by Generative Pre-Training
- GPT-2 (from OpenAI) released with the paper Language Models are Unsupervised Multi Task Learners
- Transformer-XL (from Google/CMU) released with the paper Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context
- XLNet (from Google/CMU) released with the paper XLNet: Generalized Autoregressive Pre Training for Language Understanding
- XLM (from Facebook) released together with the paper Cross-lingual Language Model Pretraining

Here is a nice tutorial on how to use PyTorch- Transformers: Introduction to PyTorch-Transformers: An Incredible Library for State-of-the-Art NLP (with Python code)

# The Steps required to build a Natural Language Processing Model

There are majorly three steps in building an NLP based Model: preprocessing the dataset, generating text-based embeddings or numerical representations and building the model. Let's have a closer look at these!

## A) Preprocessing

The preprocessing steps can be divided into four major steps: Cleaning, Tokenization ,Normalization, and Stopwords removal

### 1. Cleaning

Here are some of the most frequent types of noise that is present in text data:

#### a. HTML tags

When working with documents that have been downloaded/ scraped from the internet, you will encounter situations where you want to extract the text present among the HTML tags.

```
<body bgcolor='#333333'>
<blockquote>
  <h1><font color="#FFFF00">WebPicSet Application</font></h1>
  <p> <font color="#00FF00">Program by Mike Strong</font></p>
  <h4>Origins</h4>
  <p>WebPicSet began in 1997, before the Windows Viewer in XP, as a utility to
    create low resolution files for quick viewing in order to determine which
    pictures I wanted to select for more work within Photoshop. I wrote this in
    VisualBasic-6 using a graphics library I purchased. Before long it morphed
    into a utility to created web-sized photo files and thumbnails automatically
    rather than creating each one individually in Photoshop (before Photoshop
    included a batch utility for this purpose). </p>
  <p>I kept doing minor patches until in 2009 the graphics library I was using
    would not function in 64-bit Vista, or any Vista. Further, the library was
    getting too slow as the newer environments began chunking the old functions.
    After pricing the available graphics libraries I decided they were much too
    expensive for a no-frills utility just for myself. At that point I decided
    to develop a substitute, from scratch, in Visual Basic Express, available
    freely on the web, using the built-in system commands for graphics.</p>
  <p>I also decided to incorporate a database file-catalog output (as CSV files
    for import to databases). This would function in much the same manner as ano
    utility I had written to catalog my video and still-photo files, but would
    be generated at the same time I generated thumbnail and proof pages.</p>
```

This can be handled by a lot of ways including but not limited to using Regular Expressions that can ignore all the HTML tags.

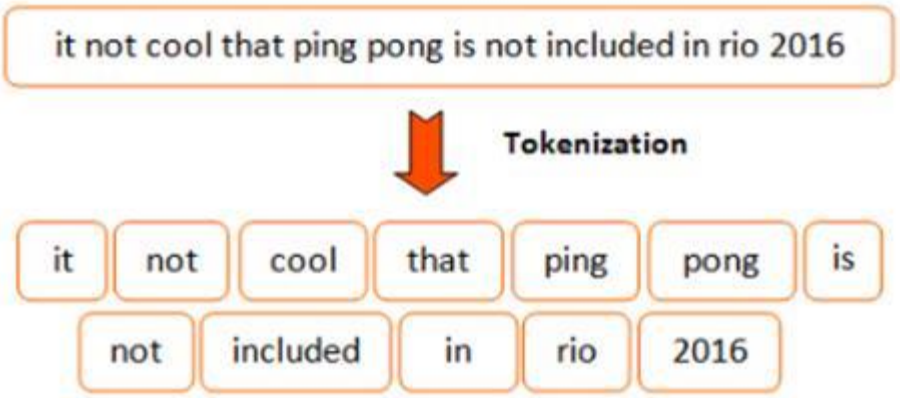
b. **Unicode and other symbols**
This form of noise is common when working with international or scientific documents, many symbols that are not part of the regular English or the ASCII system get mixed in with the text data and it is hard to make sense of these symbols in NLP systems.


These symbols belong to the Unicode system and can be dealt with both using Regular Expressions and the Unicode related functions that are available for strings in python.

c. **Removing numbers**
Numbers aren't always good, especially in case of NLP systems numbers do not add much meaning to a given piece of text and so are usually filtered out.

d. **Links**
Links can be of many forms and most of them consist of strange symbols or short-codes that can be present in your document if you have scraped it from the internet.

## 2. Tokenization




Tokenizing means splitting your text into minimal meaningful units. It is a mandatory step before any kind of processing. Like in the given example, the sentence "it is not cool that ping pong is not included in Rio 2016" is split into individual units that make sense to the sentence. In this particular example, those units were words but there might be situations where multiple words form single token and those are known as n-grams.

## 3. Normalization

An important type of textual noise is about the multiple representations exhibited by a single word. For example – “play”, “player”, “played”, “plays” and “playing” are the different variations of the word – “play”, though they mean different things, contextually they all are similar. The step converts all the disparities of a word into their normalized form (also known as lemma) known as Normalization. Normalization is a pivotal step for feature engineering with text as it converts the high dimensional features (N different features) to the low dimensional space (1 feature), which is an ideal task for any ML model.

### (a) Stemming

Stemming is the process of converting the words of a sentence to its non- changing portions.


Types of Stemmers

You're probably confused, how can you convert a series of words to its stems. Luckily, NLTK has a few built-in and established stemmers available for you to use! They work slightly differently since they follow different rules - which you use depends on whatever you happen to be working on. Here are the tokens that we want to run our experiments on:
tokens = ['omg', ',','natural','language','processing','is','so','cool','and','i',"'m",'really','enjoying','this','workshop','!']

First, let's see the output of the **Lancaster Stemmer**:
['omg', ',', 'nat', 'langu', 'process', 'is', 'so', 'cool', 'and', 'i', "'m", 'real', 'enjoy', 'thi', 'workshop', '!']

Secondly, we try the **Porter Stemmer**:
['omg', ',', 'natur', 'languag', 'process', 'is', 'so', 'cool', 'and', 'i', "'m", 'realli', 'enjoy', 'thi', 'workshop', '!']

Notice how "natural" maps to "natur" instead of "nat" and "really" maps to "realli" instead of "real" in the last stemmer.

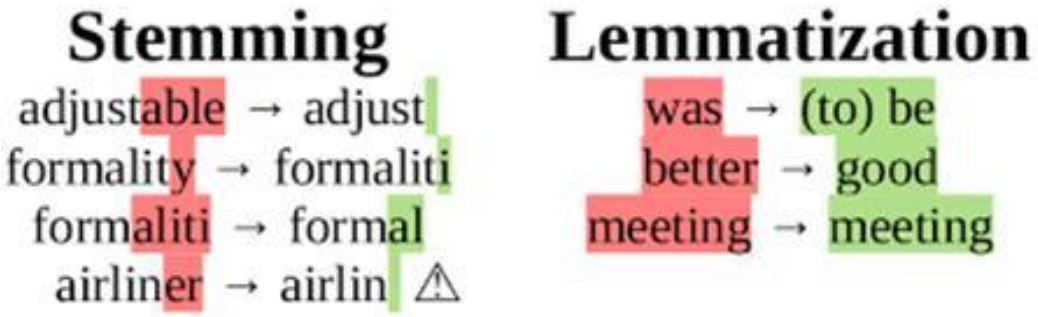
### (b) Lemmatization

Lemmatization is the process of converting the words of a sentence to its dictionary form. For example, given the words amusement, amusing, and amused, the lemma for each and all would be "amuse".

Notice the difference between Stemming and Lemmatization in the above image. Stemming is faster than Lemmatization but at the same time breaks tokens in strange ways many of which can be totally wrong. While Lemmatization is a slow

process, it correctly breaks a token into its dictionary root word or "lemma".

Let's see an example of Lemmatization in real life, here we want to try the **WordNetLemmatizer** on the following sentence
"Women in technology are amazing at coding"
This is the output after the Lemmatization:
['woman', 'in', 'technology', 'are', 'amazing', 'at', 'coding']
Notice that "women" is changed to "woman"!




4. **Stopwords**

Removing stopwords is considered as an essential preprocessing step for most NLP applications and the main reason for that is that these stopwords generally do not add new information to the text but are just a language construct. When you are trying to build an application for say, text classification then it is better to remove the stopwords before building the features. Let's see this with an example!
Suppose you want to create a text-based feature by taking into account the frequency of words in a document then this is how your feature vectors would look like before removing stopwords.

| | Sam | waited | for | the | train | was | late |
|---|---|---|---|---|---|---|---|
| Sam waited for the train | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| the train was late | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

Let's see how these vectors would change after removing the stopwords,

| | Sam | waited | train | late |
|---|---|---|---|---|
| Sam waited ~~for~~ ~~the~~ train | 1 | 1 | 1 | 0 |
| ~~the~~ train ~~was~~ late | 0 | 0 | 1 | 1 |

As you must have noticed that after removing stopwords it was much easier to differentiate between the two documents because now only the unique words were present. You can read more about text preprocessing and cleaning here: [Steps for effective text data cleaning (with case study using Python)]

## B) Embeddings and Representations

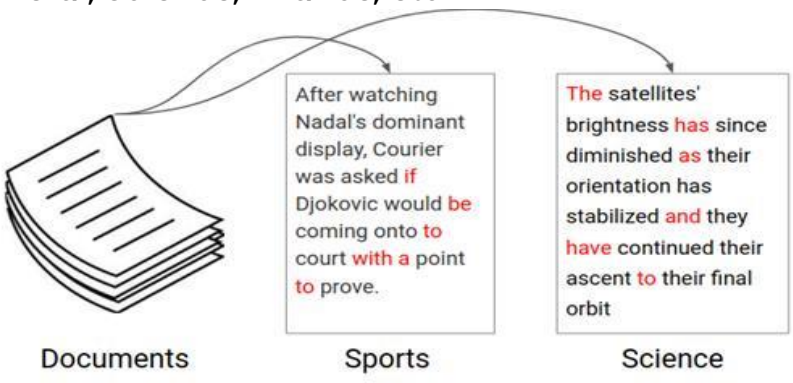
Once you are done with the basic preprocessing steps and have cleaned the dataset well the next logical step is to convert the text of the dataset into some kind of numerical representation this is because of the fact that machine learning and deep learning algorithms only work with numbers. Here are some of the most fundamental and widely used techniques to represent text data:

1. **Bag of Words (BOW)**
   Bag of Words or BOW is a technique to represent text documents by creating vectors based on the frequency of words in each document. Here is how a Bag of Words looks like:

| | Sam | waited | train | late |
|---|---|---|---|---|
| Sam waited ~~for~~ ~~the~~ train | 1 | 1 | 1 | 0 |
| ~~the~~ train ~~was~~ late | 0 | 0 | 1 | 1 |

   Notice that we are not counting the "stopwords" for better vectors. It is a very simple concept yet works really well in practice because if you are able to capture the distribution of words in each document, you are implicitly capturing its context and that's what helps in representing that document.

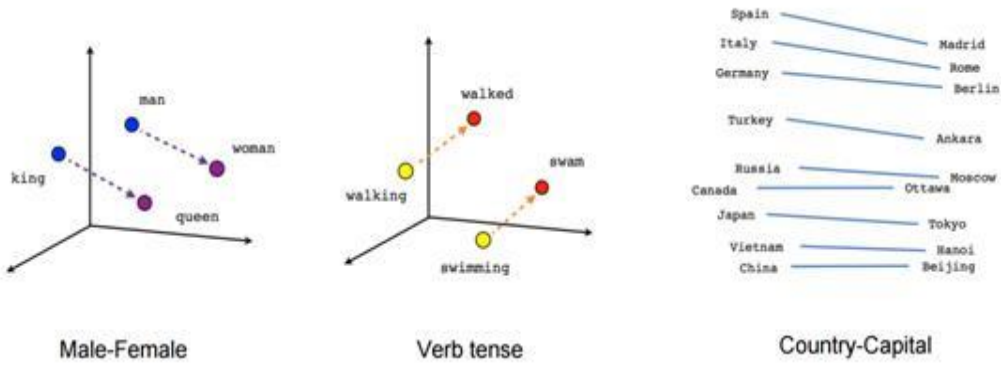
2. **Term Frequency-Inverse Document Frequency (TF-IDF)**
   Term Frequency - Inverse Document Frequency or TF-IDF as it is commonly known as is a technique of representing each document in such a way that the words that are unique to each document are given higher importance than the words that are not so rare among the documents. To understand why TF-IDF is a good idea let's take an example, say you have to classify news articles into different categories like sports, general, science, finance, etc.



- TF-IDF score is high for terms that appear quite often in a document but are not present in most of the other documents.
- TF-IDF score is lower for terms that are occurring frequently in most of the documents in a corpus. Example - Stop words (“is", “the", “a", “of", etc.)

You can learn more about TFIDF here:
An Intuitive Understanding of Word Embeddings: From Count Vectors to Word2Vec

3. **Word2Vec**
Word2Vec is a set of embeddings that are obtained by Google's Pre-trained model that has been trained on 300 million words from the Google News corpus. In this embedding, each word is represented in a 300-dimensional space and because we extend each word in such high dimensional space, we are able to capture subtle contextual relationships between words.

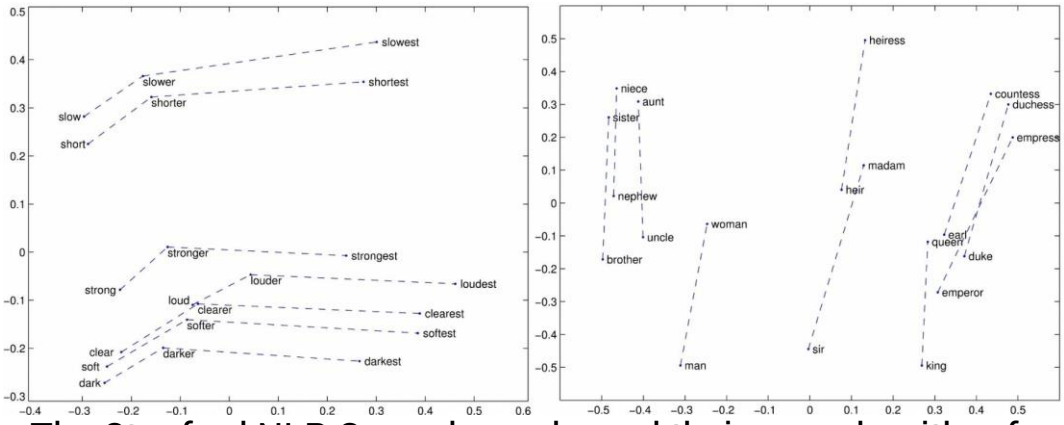



Male-Female  Verb tense  Country-Capital

In the above diagram, you'd see that if you take the word2vec of the words king, queen, man and woman then you could simply do linear algebra among words. Something like this:king - man + woman --> queen

Here is a good intro to Word2Vec embeddings:
An Intuitive Understanding of Word Embeddings: From Count Vectors to Word2Vec

## Glove




The Stanford NLP Group has released their own algorithm for training word embedding like Word2Vec and it's called Global Vectors for Word Representation or GloVe for short.

## Comparing GloVe vs Word2Vec

The following are major differences between

- GloVe and Word2Vec models - GloVe works similarly as Word2Vec. While Word2Vec is a predictive model that predicts context given a word (or vice-versa based on if you are using skip-gram or CBOW variant),
- GloVe learns by constructing a co-occurrence matrix (words x context) that basically counts how frequently a word appears in a context. Since it's going to be a gigantic matrix, we factorize this matrix to achieve a lower-dimension representation. There's a lot of details that go in GloVe but that's the rough idea.
- In word2vec, skip-gram models try to capture co-occurrence one window at a time.
- In Glove it tries to capture the counts of overall statistics how often it appears.

In practice, the main difference is that GloVe embeddings work better on some data sets, while word2vec embeddings work better on others. They both do very well at capturing the

semantics of analogy, and that takes us, it turns out, a very long way toward lexical semantics in general.

### ELMo

| Task | Previous SOTA | | ELMo + Baseline |
|---|---|---|---|
| SQuAD | SAN | 84.4 | 85.8 |
| SNLI | Chen et al (2017) | 88.6 | 88.7 +/- 0.17 |
| SRL | He et al (2017) | 81.7 | 84.6 |
| Coref | Lee et al (2017) | 67.2 | 70.4 |
| NER | Peters et al (2017) | 91.93 +/- 0.19 | 92.22 +/- 0.10 |
| Sentiment (5-class) | McCann et al (2017) | 53.7 | 54.7 +/- 0.5 |

ELMo is a novel way to represent words in vectors or embeddings. These word embeddings are helpful in achieving state-of-the-art (SOTA) results in several NLP tasks:
ELMo representations are:

- Contextual: The representation for each word depends on the entire context in which it is used.
- Deep: The word representations combine all layers of a deep pre-trained neural network.
- Character based: ELMo representations are purely character-based, allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens unseen in training.

NLP scientists globally have started using ELMo for various NLP tasks, both in research as well as the industry. Know more: A Step-by-Step NLP Guide to Learn ELMo for Extracting Features from Text

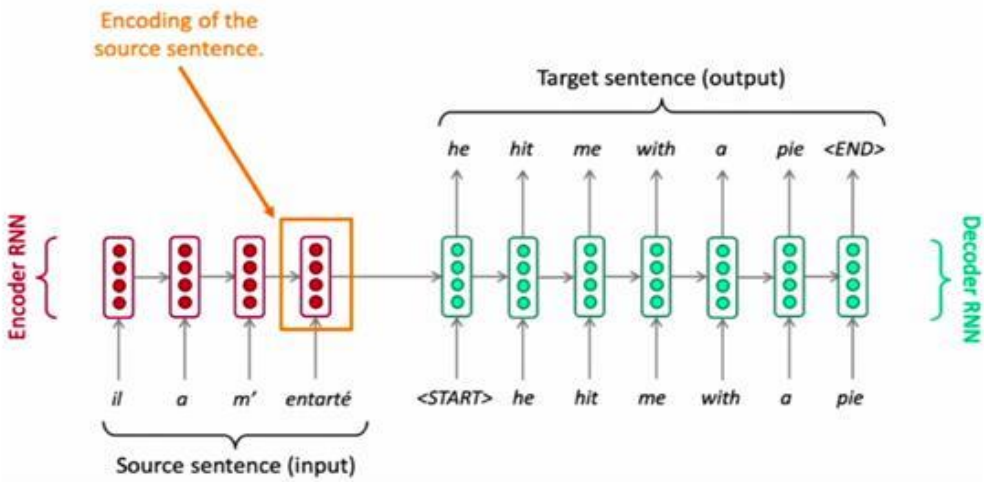
## C) Modelling Techniques

Once you have a numerical representation for your text documents the next step is to build a machine learning or

deep learning model. Here are some models that you can work with:

- **Classical ML Algorithms**
  There are many classical machine learning models like Support Vector Machines (SVM), Naive Bayes (NB), Random Forest (RF) that work really well with text data for various tasks such as sentiment classification and document classification etc.
  You can know more about some of these classical machine learning algorithms here: Essentials of Machine Learning Algorithms (with Python and R Codes)

- **Recurrent Neural Networks (RNNs)**
  Apart from machine learning based algorithms, there have been many useful deep learning algorithms that have shown to work well with text data and RNNs are one of them. Here are two of the most useful types of RNNs:
  a) Long Short Term Memory (LSTM)
  b) Gated Recurrent Unit (GRU)
  Both of these architectures have been used to solve some of the most prominent problems in NLP like Neural Machine Translation:



# The Latest Breakthroughs and Developments in Natural Language Processing

This is a great time to be working in the field of NLP. We can trace this recent rise in breakthroughs to the advent of the Transformer architecture in 2017 - a truly watershed moment. We chart the rise of NLP frameworks starting from the pioneering Transformer right up to the latest state-of-the-art architectures and libraries. Some of them include-

**Attention is All You Need (June 2017)**

- Introduced the Transformer architecture that totally changed the landscape of NLP.
- Broke existing SOTA records being held by RNN based architectures.

**Universal Language Model Fine-tuning (ULMFIT) (May 2018)**

- Successfully introduced transfer learning in NLP.
- A single universal language model trained on the WikiText103 dataset can be fine-tuned to any NLP task.
- Achieved great results even with small datasets.

**Bidirectional Encoder Representations from Transformers (BERT) (Nov. 2018)**

- First to apply the bi-directional training of the Transformer architecture for NLP tasks.
- Achieved State-of-the-Art results on the majority of NLP tasks.
- Introduced Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

**Google's Transformer-XL (Jan. 2019)**

- Overcome the limitations of Transformer

**Open AI's GPT-2 (Feb. 2019)**

- First large scale Language Model that was able to generate coherent paragraphs of text.

- Achieved state-of-the-art performance on many language modeling benchmarks.

**XLNet (Jun. 2019)**

- Used Auto-Regressive methods to overcome the limitations of BERT.
- Integrated ideas from Transformer-XL for pre-training.

**PyTorch-Transformers (Jul. 2019)**

- Provided state-of-the-art pre-trained models for BERT, Transformer-XL, XLNet etc. in python

**Baidu's Enhanced Representation through kNowledge IntEgration (ERNIE) 2.0 (Jul. 2019)**

- Applied multitask learning and a series of pretraining tasks such as capital letter predictions (since capitalized words often contain proper nouns) and tasks learning relationships between sentences.
- Outperformed high-ranking NLP models in tasks like sentiment analysis on movie reviews (SST-2) or the ability to infer meaning from a sentence (MNLI)

**Spacy-PyTorch-Transformers (Aug. 2019)**

- Implemented the PyTorch-Transformers library to utilize SOTA models for language processing.
- Made the Transformer architecture easier to deploy.

# Various jobs in Natural Language Processing

As NLP is an evolving field, new positions pop up every now and then but here are three major areas of work when working in NLP:

### NLP Engineer/Developer

These are the guys who have a solid computer engineering background and can create scalable NLP solutions/products and deploy them for the users. They are expected to be well versed with all the major frameworks and toolkits related to Natural Language Processing and Machine Learning. Here is a sample job description for an NLP Engineer:

-Design, Implement and deploy full-stack Natural Language Processing solutions on TensorFlow/PyTorch/Keras/CNTK Framework/Torch7.
- Investigate and solve exciting and difficult challenges for natural language processing through Machine learning and Deep-Learning
- Research and develop scalar natural language processing and machine learning solutions to hard problems.
- Use the latest advances in neural networks and machine learning to bring new experiences to our customers.

### NLP Scientist/Researcher

These are the people who have a strong research background in NLP or Linguistics or in general and do fundamental research in Natural Language Processing. But since they work in the industry, their research is slightly inclined towards the real-world applications of NLP. Here is some sample job description for an NLP Researcher:

- Develop scalar language models from text data to "understand" text utterances on messaging apps (WhatsApp, FB Messenger) for speakers of regional languages.
- Develop a conversational framework for training and support content.
- Develop mini-dialogue systems for common conversational use cases.

As can be seen from the above, NLP Researchers do fundamental research into areas of NLP that can be directly applied to real-world problems. Here is another one:

### NLP Data Scientist

These are the people who get the best of both worlds- NLP and Data Science. These people are not necessarily experts in the field of NLP but have enough experience to work with large amounts of unstructured data and have a strong understanding of how data can influence business decisions. Here is some sample job description for an NLP Data Scientist:

As a natural Language Processing (NLP) Data Scientist, you will specialize in processing and analyzing unstructured text. You will own the process of text analysis from beginning to end, including the following items:

- Collect and integrate unstructured data from various data sources (social, news, documents, etc.)
- Use cutting edge NLP techniques (tokenization, entity detection, lemmatization, vectorization, content classification from these unstructured text data sources texts
- Extract knowledge and patterns from unstructured texts and generate actionable business insights in a structured manner. Examples include: keywords, entities, sentiments etc.

# How you can build a Career in Natural Language Processing

Now you are asking the correct question! Given the shortage of talent in this domain, it definitely makes sense to look at building a career in natural language processing and machine learning. But before you decide, you should keep the following things in mind:

- You would need to be comfortable with coding in order to build a career as an NLP engineer or scientist.
- You do not need a background or a Ph.D. in mathematics or linguistics. You can always pick up the things you need. If you are from this background – it helps, but it is not mandatory
- For those of you transitioning from any other domain or field – plan for at least 18 months of transition.

If you get a break before – consider this as a bonus. If you are ready to build a career in natural language processing after reading the tips above you can do the following:

- Take MOOCs or University Courses
    - Coursera
    - Analytics Vidhya Courses
- Attend Meetups, Webinars, Conferences
- Solve problems yourself and learn along the way
- Become part of data science communities and learn from experts

# About the Author


Analytics Vidhya provides a community based knowledge portal for Analytics and Data Science professionals. The aim of the platform is to become a complete portal serving all knowledge and career needs of Data Science Professionals.