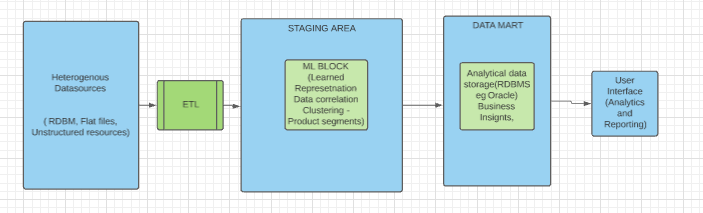
Data Flow Pipeline

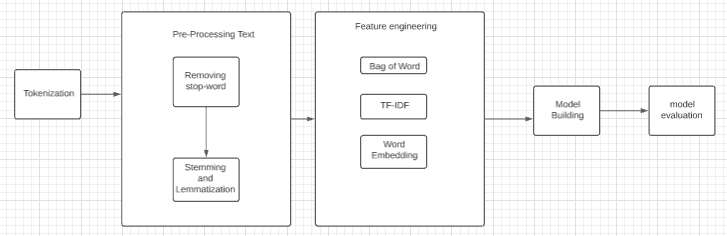


1. To begin with we extract the raw data from the Client data source using an Extraction (**ETL**) process.

The Client DBs could be any one among; RDBMS, FLAT Files, Unstructured data or NOSQL DBs.

1. This data is then moved to a **DATA Staging Area**.
2. A staging area simplifies data cleansing and consolidation for operational method coming from multiple source systems, especially for enterprise data warehouses where all relevant data of an enterprise is consolidated.
3. At this layer of the architecture ML algorithms are applied to the input data sources.
4. In our case the algorithms are applied to the Reviews section of various applications to extract feature specific reviews. The negative and positive sentiments within the reviews are segregated and loaded into a Data Base (Data Mart).
5. A data mart is a segment of a data warehouses that can provide information for reporting and analysis on a section, unit, department, or operation in the company. These broadly provide Business insights of the Data.
6. The data from the Staging DB is then used by Analytical tools to create Visualizations using the Business insights available in the processed DB.

This is the final block in the pipeline. An interactive user Interface is provided to make reading of insights as dynamic as possible for the end user.



**Tokenization**

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph.

**Sentence Tokenization**

Sentence tokenizer breaks text paragraph into sentences.

from nltk.tokenize import sent\_tokenize

text="""Hello Mr. Smith, how are you doing today? The weather is great, and city is awesome.

The sky is pinkish-blue. You shouldn't eat cardboard"""

tokenized\_text=sent\_tokenize(text)

print(tokenized\_text)

['Hello Mr. Smith, how are you doing today?', 'The weather is great, and city is awesome.', 'The sky is pinkish-blue.', "You shouldn't eat cardboard"]

Here, the given text is tokenized into sentences.

**Word Tokenization**

Word tokenizer breaks text paragraph into words.

from nltk.tokenize import word\_tokenize

tokenized\_word=word\_tokenize(text)

print(tokenized\_word)

['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'weather', 'is', 'great', ',', 'and', 'city', 'is', 'awesome', '.', 'The', 'sky', 'is', 'pinkish-blue', '.', 'You', 'should', "n't", 'eat', 'cardboard']

**Frequency Distribution**

from nltk.probability import FreqDist

fdist = FreqDist(tokenized\_word)

print(fdist)

<FreqDist with 25 samples and 30 outcomes>

fdist.most\_common(2)

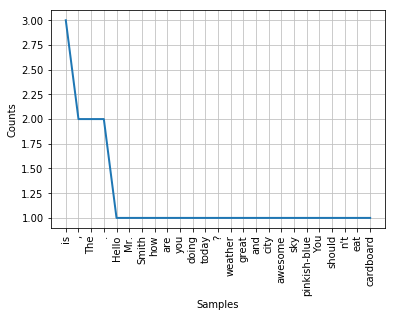
[('is', 3), (',', 2)]

# Frequency Distribution Plot

import matplotlib.pyplot as plt

fdist.plot(30,cumulative=False)

plt.show()



**Stopwords**

Stopwords considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc.

In NLTK for removing stopwords, you need to create a list of stopwords and filter out your list of tokens from these words.

from nltk.corpus import stopwords

stop\_words=set(stopwords.words("english"))

print(stop\_words)

{'their', 'then', 'not', 'ma', 'here', 'other', 'won', 'up', 'weren', 'being', 'we', 'those', 'an', 'them', 'which', 'him', 'so', 'yourselves', 'what', 'own', 'has', 'should', 'above', 'in', 'myself', 'against', 'that', 'before', 't', 'just', 'into', 'about', 'most', 'd', 'where', 'our', 'or', 'such', 'ours', 'of', 'doesn', 'further', 'needn', 'now', 'some', 'too', 'hasn', 'more', 'the', 'yours', 'her', 'below', 'same', 'how', 'very', 'is', 'did', 'you', 'his', 'when', 'few', 'does', 'down', 'yourself', 'i', 'do', 'both', 'shan', 'have', 'itself', 'shouldn', 'through', 'themselves', 'o', 'didn', 've', 'm', 'off', 'out', 'but', 'and', 'doing', 'any', 'nor', 'over', 'had', 'because', 'himself', 'theirs', 'me', 'by', 'she', 'whom', 'hers', 're', 'hadn', 'who', 'he', 'my', 'if', 'will', 'are', 'why', 'from', 'am', 'with', 'been', 'its', 'ourselves', 'ain', 'couldn', 'a', 'aren', 'under', 'll', 'on', 'y', 'can', 'they', 'than', 'after', 'wouldn', 'each', 'once', 'mightn', 'for', 'this', 'these', 's', 'only', 'haven', 'having', 'all', 'don', 'it', 'there', 'until', 'again', 'to', 'while', 'be', 'no', 'during', 'herself', 'as', 'mustn', 'between', 'was', 'at', 'your', 'were', 'isn', 'wasn'}

**Removing Stopwords**

filtered\_sent=[]

for w in tokenized\_sent:

if w not in stop\_words:

filtered\_sent.append(w)

print("Tokenized Sentence:",tokenized\_sent)

print("Filterd Sentence:",filtered\_sent)

Tokenized Sentence: ['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?']

Filterd Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?']

**Lexicon Normalization**

Lexicon normalization considers another type of noise in the text. For example, connection, connected, connecting word reduce to a common word "connect". It reduces derivationally related forms of a word to a common root word.

**Stemming**

Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes. For example, connection, connected, connecting word reduce to a common word "connect".

# Stemming

from nltk.stem import PorterStemmer

from nltk.tokenize import sent\_tokenize, word\_tokenize

ps = PorterStemmer()

stemmed\_words=[]

for w in filtered\_sent:

stemmed\_words.append(ps.stem(w))

print("Filtered Sentence:",filtered\_sent)

print("Stemmed Sentence:",stemmed\_words)

Filtered Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?']

Stemmed Sentence: ['hello', 'mr.', 'smith', ',', 'today', '?']

**Lemmatization**

Lemmatization reduces words to their base word, which is linguistically correct lemmas. It transforms root word with the use of vocabulary and morphological analysis. Lemmatization is usually more sophisticated than stemming. Stemmer works on an individual word without knowledge of the context. For example, The word "better" has "good" as its lemma. This thing will miss by stemming because it requires a dictionary look-up.

#Lexicon Normalization

#performing stemming and Lemmatization

from nltk.stem.wordnet import WordNetLemmatizer

lem = WordNetLemmatizer()

from nltk.stem.porter import PorterStemmer

stem = PorterStemmer()

word = "flying"

print("Lemmatized Word:",lem.lemmatize(word,"v"))

print("Stemmed Word:",stem.stem(word))

Lemmatized Word: fly

Stemmed Word: fli

**POS Tagging**

The primary target of Part-of-Speech(POS) tagging is to identify the grammatical group of a given word. Whether it is a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc. based on the context. POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word.

sent = "Albert Einstein was born in Ulm, Germany in 1879."

tokens=nltk.word\_tokenize(sent)

print(tokens)

['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',', 'Germany', 'in', '1879', '.']

nltk.pos\_tag(tokens)

[('Albert', 'NNP'),

('Einstein', 'NNP'),

('was', 'VBD'),

('born', 'VBN'),

('in', 'IN'),

('Ulm', 'NNP'),

(',', ','),

('Germany', 'NNP'),

('in', 'IN'),

('1879', 'CD'),

('.', '.')]

POS tagged: Albert/NNP Einstein/NNP was/VBD born/VBN in/IN Ulm/NNP ,/, Germany/NNP in/IN 1879/CD ./.

**Sentiment Analysis**

Nowadays companies want to understand, what went wrong with their latest products? What users and the general public think about the latest feature? You can quantify such information with reasonable accuracy using sentiment analysis.

Quantifying users content, idea, belief, and opinion is known as sentiment analysis. User's online post, blogs, tweets, feedback of product helps business people to the target audience and innovate in products and services. Sentiment analysis helps in understanding people in a better and more accurate way. It is not only limited to marketing, but it can also be utilized in politics, research, and security.

Human communication just not limited to words, it is more than words. Sentiments are combination words, tone, and writing style. As a data analyst, It is more important to understand our sentiments, what it really means?

There are mainly two approaches for performing sentiment analysis.

* Lexicon-based: count number of positive and negative words in given text and the larger count will be the sentiment of text.
* Machine learning based approach: Develop a classification model, which is trained using the pre-labeled dataset of positive, negative, and neutral.

In this Tutorial, you will use the second approach(Machine learning based approach). This is how you learn sentiment and text classification with a single example.

**Text Classification**

Text classification is one of the important tasks of text mining. It is a supervised approach. Identifying category or class of given text such as a blog, book, web page, news articles, and tweets. It has various application in today's computer world such as spam detection, task categorization in CRM services, categorizing products on E-retailer websites, classifying the content of websites for a search engine, sentiments of customer feedback, etc. In the next section, you will learn how you can do text classification in python.