1. METHODOLOGY SELECTED

As methodology deals with the various approaches that we used to collect the data and organising it the required so that people can access it quickly. So first of all, we started collecting the information related to various movies like on the basis of the actor and the actress and their releasing year which also involves movie ratings. Moving towards the concepts of AI used are Count Vectorizer , NLTK packages , bag of words model etc. we also used various diagrammatic to attract the users at the time of viewing the status of the movies.

Collection of data involves various types movie posters and the introduction images and the organising the movies based upon the action, fiction, Enjoyment related movies in the table. We also used csv\_files to place our collected data in the manner such that we program it quickly. And finally, to the last we also used some online references to collect the ratings of movies. To the topics that we used in the developing the code includes the various fields. Those are:

COUNTVECTORISER:

Count Vectorizer is used to transform a corpus of text to a vector of term / token counts. It also provides the capability to pre-process your text data prior to generating the vector representation making it a highly flexible feature representation module for text. It is a way of extracting features from the text for use in machine learning algorithms. The process of converting NLP text into numbers is called vectorization in ML. Different ways to convert text into vectors are: Counting the number of times each word appears in a document.

* It is a set of tools used so computers can understand language like humans do.
* Gaining insights from your text
* Taking text and converting them into features to put into your model

“Where is natural language processing being currently used?”

* There are certainly many use cases for it such as: Google searches, voice recognition (i.e. Siri, Alexa, Bixby), Autocorrect, and sentiment analysis for Twitter
* Natural Language Processing allows us to convert the text we want to analyse and reprocess them into features to be put into our model. There are methods that can be used in the Natural Language Processing toolbox that can be used to pre-process our text: Count Vectorization and Term Frequency-Inverse Document Frequency. Count Vectorization involves counting the number of occurrences each word appears in a document (i.e. distinct text such as an article, book, even a paragraph!). Python’s Sci-kit learn library has a tool called Count Vectorizer to accomplish this.

NLTK PACKAGES

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux.

Best of all, NLTK is a free, open source, community-driven project. NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language. ”NLP with python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analysing linguistic structure, and more.

NLTK is a leading platform for building Python programs to work with human language data. . Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analysing linguistic structure, and more.

BAG OF WORDS

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

It works like this: Create a bucket for each unique word you want represented (the vocabulary). Next go over the text and put a token in the right buckets for the words you encounter. You can build this with plain python, but as I mentioned before it is not the most efficient method.

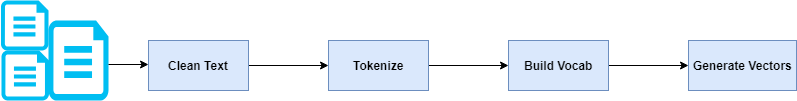
Bag of Words (BOW) is a method to extract features from text documents. These features can be used for training machine learning algorithms. It creates a vocabulary of all the unique words occurring in all the documents in the training set.

In simple terms, it’s a collection of words to represent a sentence with word count and mostly disregarding the order in which they appear.

BOW is an approach widely used with:

1. Natural language processing
2. Information retrieval from documents
3. Document classifications

On a high level, it involves the following steps.



Generated vectors can be input to your machine learning algorithm.

Insights into bag of words

The BOW model only considers if a known word occurs in a document or not. It does not care about meaning, context, and order in which they appear.

This gives the insight that similar documents will have word counts similar to each other. In other words, the more similar the words in two documents, the more similar the documents can be.

Limitations of BOW

1. Semantic meaning: the basic BOW approach does not consider the meaning of the word in the document. It completely ignores the context in which it’s used. The same word can be used in multiple places based on the context or nearby words.
2. Vector size: For a large document, the vector size can be huge resulting in a lot of computation and time. You may need to ignore words based on relevance to your use case.
3. This was a small introduction to the BOW method. The code showed how it works at a low level. There is much more to understand about BOW. For example, instead of splitting our sentence in a single word (1-gram), you can split in the pair of two words (bi-gram or 2-gram). At times, bi-gram representation seems to be much better than using 1-gram. These can often be represented using N-gram notation. I have listed some research papers in the resources section for more in-depth knowledge.

STOP WORDS:

Here stop words is the other important topic that we used int eh development of the code. these words are commonly used in English language to reduce the importance of unimportant words allowing the applications to use the important words instead. These stop words help us to acquire the good results with in the short period of time as we are totally concentrating on the important words. stop word is a commonly used word (such as "the") that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. When building the index, most engines are programmed to remove certain words from any index entry. The list of words that are not to be added is called a stop list. Stop words are deemed irrelevant for searching purposes because they occur frequently in the language for which the indexing engine has been tuned.

You do not have to code BOW whenever you need it. It is already part of many available frameworks like Count Vectorizer in sci-kit learn.

PROJECT DESCRIPTION:

Firstly, I analysed the dataset and tried to find to most relevant attributes. For example, the "number of rating" attribute is not so relevant in my opinion, because the dataset contains a lot of popular movies without any rating. To find the best solution, I decided to develop an algorithm that can be fine-tuned with weights for each used attribute.

#### The final list of applied attributes is:

* Average user ratings
* Number of user ratings
* Genres similarity
* User given tags similarity
* Title similarity
* Movie year