**NATURAL LANGUAGE PROCCESSING REPORT**

**Natural language processing** refers to the branch of artificial intelligence, which allows machines to understand and process human language. It is converting given inputs such as words or speech etc into numbers that a machine can understand and predict outputs of.

There are various applications of NLP techniques in everyday life such as a sentiment analyser that can help brands detect the feelings its customers have towards a product based on their comments.

This all works based on an **NLP Pipeline** which is a set of processing elements connected in series and is as follows:

* **Segmentation**- It is the process of dividing a sentence into its component sentences. For example- “I went to the golf course, it was a lot of fun.” After segmenting this it would be two separate sentences- 1) “I went to the golf course”. 2) “It was a lot of fun.”
* **Tokenization**- This involves dividing sentences into its constituent words. Using the previous example, “I went to the golf course” becomes- “I” “went” “to” “the” “golf” “course”. Furthermore, another step in this process to make processing the data easier for a model includes removing the “stop words” This includes sentence connectors, grammar etc, words that would not affect the classification of data. E.g.- “I:” “to” etc.
* **Stemming**- Stemming involves obtaining the stem of a word. New words can be found by removing the affixes and interchanging. Post removing the said affixes, we get the “root stem” of the given word. For example, skip is the root stem, whereas skipped, skipping, skipper are all the words with affixes of “ed”, “er” “ing” attached.
* **Lemmatization-** This procedure aims to derive the meaning of a word and return a valid word that exists in the language. For example, the lemma of the words went, going and gone are “go”.
* **Part of Speech Tagging (POS)** - Part of Speech Tagging or POS identifies which component of speech a word belongs to. Basically, classifying what a word is classified as such as a noun, verb, pronoun etc.

If we take a sentence “The lemonade quenched her thirst” Under POS this sentence would be classified as:

|  |  |
| --- | --- |
| The | Determiner |
| Lemonade | Noun |
| Quenched | Verb |
| Her | Pronoun |
| Thirst | Noun |

* **Named Identity Recognition** – It is the process of classifying words into subcategories making it easier to categorize these include: 1) Person 2) Quantity 3) Location 4) Organisation 5) Movie 6) Monetary Value.

When studying about NLP it made it easier for me to learn about the concepts based on the different techniques that can be implemented, according to what goal you are trying to achieve. Specific techniques become more suitable. There are three different approaches to NLP:

* Language And Heuristics
* Machine Learning
* Deep Learning

Language and Heuristics refers to filtering input data based on the required outcome. No prediction and learning are required. For example, filtering through an investment report for the year FY2021, but specifically requiring data for Q3. Using a tutorial from YouTube and a website called “regex101” I learned about an efficient library within Python that can be inferred to filter data smoothly and retrieve the required components. I have included below a few snippets of coding I had practiced from the tutorial whilst following along.

A screenshot of a computer

Description automatically generatedPractice:

When it comes to the other aspects of NLP, there are various types of techniques that can be implemented. They can be split into two categories Wherein Machine Learning and Deep learning are classified based on:

* Text Representation
* Words Embeddings

**BAG OF WORDS:**

Bag of Words is a text representation format wherein a vocabulary is formed based on a given corpus and vectors (count vectorizer) can be made based on the word frequency forming a bag of words that includes all unique words, post preprocessing (stemming and lemmatization) which tells you the word frequency of each unique word.

In the above example, the objective is to determine which company a given article belongs to Telsa or Apple? So looking at lets say three articles we can form a numeric representation for it this is a simplified table, used for ease of explanation, in reality a lot more words would be present in the vocabulary.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tesla | Elon | Apple |
| Article 1 | 7 | 8 | 0 |
| Article 2 | 0 | 1 | 6 |
| Article 3 | 24 | 16 | 2 |

Based off these vectors it is easy for us to classify which company a article belongs to.

A screenshot of a computer

Description automatically generatedI implemented an example for this, while following a tutorial, a dataset which includes 1000s of emails categorized as spam and not spam is implemented using a bag of words model. A Naïve Bayes classifier is used to train, and post that we are able to determine whether a mail in the given dataset is spam or not spam. Other classifiers can also be used such as KNN and Random Forest. I have added comments for ease of understanding.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Bag of words however has its drawbacks in implementation such as sparse matrices when the word count is high. Moreover, it generates its vocabulary as a bag of unordered words. This doesn’t allow it to analyse the sentence structure along with the semantics relationships between words. It can have high memory consumption, as a vector formed with a large dataset can have over 100,00 unique words. When looking at the vectors if we look at similar words such as “good” and “great” we could expect the vectors to be similar. This wouldn’t work in this case as the vector for good would have a value “1” in the good element but “0” in the great element and vice versa.

**N-gram Models –**

N-gram models come in handy in specific situations such as predicting the next word in a sentence given a large corpus. It is an improvement on BOW as it takes into account sentence structure, and its order in comparison to the order less format of the BOW model. It is a sequence of words and can be classified based on the number of words it considers at a time. For example, a unigram could be: “please” “submit” “your” homework. In this same case a bi gram could be “please submit” “your homework”. It works on probability based on a given word w2 and how many times the preceding word w1 may have occurred.

For example, predicting “the” as the next word in a sentence that says “it’s water is so transparent that” can be found by counting the sentence “it’s water is so transparent that” occurs and how many times it is followed by “it’s water is so transparent that the”

Count (the water is so transparent that the) / Count(the water is so transparent that)

*P*("word2"∣"word1")=Count("word1")Count("word1 word2")​

**TF-IDF-**

TF-IDF is a method that can help reduce the impact of certain non-necessary words when forming feature vectors, words that do not assist with our desired goal. If we look at the example from BOW, words such as markets, investors etc are words that would be added to the vocabulary but aren’t impactful in determining which company the given article belong to. This is where TF-IDF comes into play.

We take into account **“document frequency”.** This technique rather than looking at the word frequency in a specific article looks at the given word as a term “t” and analyses how many times “t” has occurred across all articles.

There are two components here:

* Inverse Document Frequency (IDF) is found using log (Total number of articles/ how many articles “t” appeared in). This can help reduce the amount of impact the term has since words like “market” or “investors” could’ve appeared across all documents whereas more integral words that act as our determiners such as “Apple” and “Elon” would’ve appeared less frequently.
* Term Frequency (TF) differs from word frequency as suppose if article 1 has 500 words and article 2 has 10000 words. The impact of word frequency becomes a moot point. Hence term frequency is used wherein the total number of words is taken into account. It is found by: total number of times “t” is present in article 1/ total number of token in article 1.

I implemented a basic version of this in python below are the screenshots of the code:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Text representations techniques such as the Bag of Words model, TF-IDF, and n gram models. TF-IDF and BOW are static representations, they do not analyse the given data for its semantic relationships and sentence structure. They form a vocabulary based on the unique words in a corpus. Whereas word embeddings are trained using neural networks such as skip-gram and CBOW. The neural networks can adapt to words given and back propagate the predicted output w.r.t to the actual output and adjust the weights accordingly.

Word embeddings are more efficient in the manner that the vectors usually span around 300, Whereas in a BOW model if there are 10,000 unique words then the vector will span 10,000 values.

The Deep Learning aspects of NLP come into play here with models such as Word2vec and GloVe vectors.

**Word2Vec-**

Word2Vec is a fascinating NLP technique wherein Neural Networks are used in order to analyse semantic relationships between words. This allows similar words such as the previously mentioned “good” and “great” to have similar vectors unlike BOW.

Word2Vec basically allows a fake problem to be created which while solving, we get word embedding that are a vector of the weights given to each word in a neural network. We can feed the words king-man+woman and the model would return queen.

Let’s say our corpus is a story on King Ashoka. Where the sentences “the mighty king Ashoka ordered his ministers” along with “the might emperor Ashoka ordered his ministers”. Using CBOW we can form a neural network wherein “ordered” and “his” are given as context and our target is to get “king” as the output, Of course, initially our network will hand us the wrong output. This can be called ‘y/’ subtracting this from the actual output “y” we can provide it back to our input as backward error propagation which will adjust its weights accordingly. This is down with the entire dataset, eventually training the model we can determine this by providing an epoch value which is how many times a dataset has entirely been fed to the network. Essentially, we create a window 🡪 form training samples 🡪 provide for example a one hot encoded word 🡪 feed in context words 🡪 do y/ minus y 🡪 adjust weights accordingly 🡪 [w1,w2,w3,w4] which are the weights of the hidden layers here we assume there are 4, this forms our word embedding for the word “king”. After training this vector will have similar values to that of the word emperor.

A screenshot of a computer

Description automatically generatedFollowing a tutorial, I coded a word2vec model using a large data set of amazon product reviews I explored the genism library further and displayed some other features too:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**GloVe Vectors:**

GloVe or global vectors derive semantic relationships between words using co-occurrence matrices. It creates a co-occurrence matrix from a large corpus based on how frequently words co-occur with each other. Taking a simple example: the sentences “I love Data Science” and “I love Learning Data” is used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i/j | I | Love | learning | Data |
| I | 0 | 2 | 0 | 0 |
| Love | 2 | 0 | 1 | 1 |
| Learning | 0 | 1 | 0 | 0 |
| Data | 0 | 1 | 0 | 0 |

Here the probability can be found by Pij = P (Wi | Wj) = Xij/Xi = P(I | Love) = 2/2.

**Conclusion:**

Each technique learned has various use cases based on what we are trying to achieve. Learning these models and the way it works makes me excited for the future when real implementation can be done. Seeing the use of this field in real time is very enriching as it helps with our quality of life and is implemented in every form , be it this very word document predicting my next sentence that has probably been trained using one of the models I learnt, to spell check within our phones, keyboard prediction and machine translation, the possibilities are endless.