Implementation of Text Representation Methods

Exercise 3 – ITE Elective

Shan Hiro Rosario   
Computer Studies and Engineering  
Mandaluyong City, Philippines   
shanhiro.rosario@my.jru.edu

***Abstract*—The proponent of the study aims to present the documentation and analysis of the Natural Language Processing (NLP) methods regarding text representation techniques, specifically to familiarize the beneficiaries of the study with various NLP methods such as Bag of Words, TF-IDF, and Word Embeddings by providing an observation, comparison analysis and direct implementation of the code including its step-by-step processes. Google Colab is the primary source of code implementation whereas the addition of markdown explanation, purpose of each block of code including its inputs and output are the key observations made which are referenced to the original context and differences of the NLP tasks used. By analyzing the performance and efficiency of the three techniques used, it is within the consideration that the insights provided would lead to the beneficiaries’ ability to comprehend deeply into how text data can be represented, converted and processed for future NLP tasks—with the sole aim of building foundational knowledge in NLP by preparing the subsets or textual examples to pave the way for advanced studies and practical applications on the industry.**

***Keywords*—Natural Language Processing, Text Representations, TF-IDF, Bag of Words, Word Embeddings, Text Analysis, Data Processing**

# **Introduction**

Natural Language Processing involves many fields, its broad and interdisciplinary industry spans several complex domain, such as Sentiment Analysis, Multimodal NLP, Conversational AI, Speech Processing and many more. The ability to handle such advanced studies and its practical applications requires the practitioner of the field to enhance their understanding of foundational knowledge in the field—In the context of NLP, these are the text representation methods such as Bag of Words, TF-IDF and Word Embeddings. A Major part of the Preprocessing pipeline involves these techniques and since computers cannot directly comprehend raw text, it is imperative to translate language into numerical or vector-based representations which are, specifically: Text Representation Methods. The ability to perform various tasks like classification, generation and translation lies in the practitioner’s ability to “train” the computers to capture the meaning, structure and context of the sentences for NLP tasks. For the matter at hand requires machines who only understand 0 and 1 to grasp the nuances of a whole language, and in this case, the internationally-used English language. [1]

1. **RELATED WORK**

Researchers Mikolov et al., introduced a word embedding methodology specifically called the Word2Vec model which uses neural networks to produce word representations based on context, in which, also demonstrates the word embeddings have been in used in the NLP tasks for decades as the research was made in 2013 [2]. We can observe the recent advances in NLP using pretrained transformers, like BERT which empirically surpassed traditional methods proved by Devlin et al., using techniques like generating dynamic, context-sensitive word representations, whereas significant improvements can be directly seen in question answering and sentiment analysis of the model as presented by the researchers[3]. Further advancement uses robust models with the ability to integrate text with image in its NLP tasks, also called the LXMERT as studied by the researchers Tan & Bansal, 2019 [4]. These sophisticated methods have been groundbreaking at the time of their development in which, at the time of this writing at the year 2025 can be directly seen it as a normalized and commercial products that also has available and free tiers such as OpenAI that posses powerful abilities including but not limited to: Content Creation & Editing, Programming Assistance, Research Assistance, Data Science & Mathematical Assistance. The Foundational knowledge of text representations then, is critical to reach the heights of these technologies performed using NLP Models.

# **Methodology**

There are Four Text Representation Methods: Dictionary Lookup, One-Hot Encoding, Bag-of-Words, and Term Frequency-Inverse Document Frequency (TF-IDF). They will be presented in order below, starting form Dictionary Lookup alongside the sample sentences or data that will be used before the conversion into a text representation technique alongside its given output.

1. Dictionary Lookup:

A screen shot of a computer code

AI-generated content may be incorrect.

Figure 1: Implementation of Dictionary Lookup

These are used to build a word dictionary mapping using numeric IDs which, is translated directly to simply extract words from the sentence variable using the tokenization concept (Similar to an array and its indexes)

* def extract\_words is the function that takes in the argument sentences
* unique\_words = set() craetes an empty set which is essential to the tokenization method to store them.
* words = sentence.split() splits the sentence based on whitespace
* unique\_words.update(words) adds tokens to the set and removes duplicates
* word\_dict = {i: word for i, word in enumerate(unique\_words)} loops over the set and enumrate to create a dictionary mapping (ex. i = word)
* return word\_dict simply  returns the mapping(word\_dict variable) so it is the output of the function extract\_words.

.

These are the list of example "dataset" that is defined in the variable sentences which assigns a three sentence string due to the "," and returns the variable as seen in the last line.are shown in the figure 1.1 below:

A white background with red text

AI-generated content may be incorrect.

Figure 1.1: Sample Data

Dictionary Lookup allows for quick “searches” of words based on the assigned numerical values. It is important to note that it doesn’t provide a direct method of text representation as vectors or matrices, only mapping a word in a key:value pair for queries.

1. One Hot Encoding

A computer screen shot of a code

AI-generated content may be incorrect.

Figure 2: Implementation of One Hot Encoding

The sentences variable are reused. However, the table 1 below provides the exact values of the variable. token\_index variable is used to assign each word a unique token (or integer) starting at 1 similar to the previous function of word\_dict earlier including its limitations. (The = 1, cat = 2, and so on.) Increments starts at 1 due to counter += 1. the token\_index.update({considered\_word : counter + 1}) assigns each word an integer.

Table 1: Sample Data for One-Hot Encoding

Note that these values are stored in an array and separated using comma (,)

|  |  |
| --- | --- |
| **First Sentence** | **Second Sentence** |
| The cat in the hat | "The dog in the house" |

1. Bag of Words

A screenshot of a computer code

AI-generated content may be incorrect.

Figure 3: Implementation of Bag of Words

CountVectorizer() and fit\_transform are the built-in methods in the library to turn the variable sentences into a bag of words representation. Then the code simply prints out the result.

Note that the numbers that is represented on the vector is how many times the "word" is repeated in the sentence. Each row represents the sentence (the cat in the hat = first row) and each column is the converted words. For example, bird appeared 0 times in the first sentence, cat appeared 1 time, dog appeared 0 time, hat appeared 0 time and in appeared twice in this format [0 1 0 1 0 1 0 2] This can be further observed in the figure 3.1 below

A white background with black text

AI-generated content may be incorrect.

Figure 3.1. Sample Output

1. TF-IDF

A computer code with black text

AI-generated content may be incorrect.

Figure 4: TF-IDF Implementation

TfidfVectorizer converts the texts in the variable documents into TF-IDF representations and fit\_transform works the same way as previously, which fits the model to  the documents and transforms them into the TF-IDF format.

A white rectangular object with black text

AI-generated content may be incorrect.  
 Figure 4.1: TF-IDF Formula

The formula in the figure 4.1 above determines the exact calculation of the Term Frequency (TF) and Inverse Document Frequency (IDF) and are automatically assigned by the sci-kit learn library using the TfidfVectorizer() as seen in the figure 4.

The Table 2 below shows the output of the TF-IDF matrix in a more readable and informative way, as its representations formed from the variable documents that contains the information will also be shown in the table 2.1

**Table 2. TF-IDF Matrix Representation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Document | and | document | first | is | one | second |
| 1 | 0.000 | 0.469 | 0.617 | 0.365 | 0.000 | 0.000 |
| 2 | 0.000 | 0.728 | 0.000 | 0.283 | 0.000 | 0.479 |
| 3 | 0.497 | 0.000 | 0.000 | 0.294 | 0.497 | 0.000 |

The formula at the top of this section is used to display the matrix where the calculated numbers are ranked based on their importance (higher is more important and lower means its a common word therefore its less important on that particular document or in this case-- sentence) The TF-IDF matrix helps us understand which words are more distinctive or important for each document in a corpus using the formula and the feature names.

# **Results**

The four Text Representation methods would be further discussed by the use of the comparison approach below:  
  
**A. Dictionary Lookup**

Used to map words to numerical values or id it is assigned to, usually formed in chronological order whereas the first few words in a sentence is modeled after 0 or 1. Essentially, it is a key-value mapping that focuses on fast lookups or extraction of words. This leads to efficient use of storage space if the dictionary is low, the memory requirements would also be relatively low. There are no training required for this method because rules are predetermined, which generates uncertainty in the method whereas it cannot analyze or determine the context of the sentence because mapping is one-to-one solely composed of such structure that inevitably involves no sentence-level understanding.

**B. Bag Of Words**

This method represents text as word frequency vectors, whereas it ignores order but keeps count of the frequency or the amount of times the word has been repeated in a sentence, leading to easy to implement as it is directly counting words, however, it is also memory-heavy in terms of its scalability as new words equals the increase of vector length that could potentially reach thousands or millions. However, as it tracks the frequency of words through a vector, its advantages comes from text classification whereas the practitioner categorizes SMS messages, documents, or emails on whether they contain “free” “win” “offer”, or URL links that are not top level domains may be classified as a spam. Additionally, bag of words could also group or categorize articles or messages into industries like “tech”, “finance”, “sports”, “politics. Check figure 3.1 for sample output.

**C. One-Hot Encoding**

One hot encoding essentially transforms categorical data like sentences that are previously used into a binary vector (0, 1) whereas 1 represents the word while 0 means empty. It observes similarities on bag of words specifically on its vectors similar to figure 3.1 however, one-hot encoding does not track the frequency of words but it is a positional-mapping of the words through a vector. See figure 5 below for observation. Note that the positioning is at the end of each sentence, whereas 1 equals ‘The’, and 2 equals ‘cat’:

A white paper with black text

AI-generated content may be incorrect.

Figure 5. One Hot Encoding Sample

A screenshot of a computer code

AI-generated content may be incorrect.

Figure 5.1 Final Representation

D. **Term Frequency Inverse Document Frequency**

The complexity of the text representation are mostly due to the IDF calculation which highlights unique words that turns into a good ranking or searching of the word, however, its calculation also reduces the weight of frequent terms, in which, the nuances of certain words might be lost in its process, opposite to the bag of words that counts the frequency of words that could be classified or categorized into various outputs. However, the nature of TF-IDF lies in its ability to scrutinize entire documents unlike methods such as Dictionary Lookup and One Hot Encoding which solely focuses on the word itself, which turns into more suitable on handling sentences in details and its contexts at a smaller level, TF-IDF highlights important words in a document, specifically its unique keywords and downplays common words which are robust implementations on search engines, document similarity or checker, keyword extraction, and text classification as well. Table 6 below shows the calculation of the TF-IDF weight whereas higher numbers mean its more important (unique) and lower means its more common in the sample output of Figure 6.1

A screenshot of a white sheet with numbers

AI-generated content may be incorrect.

Figure 6. TF-IDF Sample Output.

A close up of red text

AI-generated content may be incorrect.

Figure 6.1. Sample Data

##### **V. Discussion**

Text Representation Methods are simple techniques that turns sentences or texts in a structured format when it is converted or transformed. Dictionary Lookup and One-Hot Encoding are the simplest—A positional mapping and key-value pair extraction. Recommended for small vocabulary or set of data, however, it doesn’t provide the number of times a word has been present in a sentence like Bag of Words. TF-IDF meanwhile, gives weights to words that are unique and important across multiple documents of which, are capabilities outside of the three mentioned before as scalability is an issue for those and TF-IDF is beneficial for large sets of data for example, search engines and multiple documents. These are the methods to transform text into machine-readable format whereas each of them has its own capabilities depending on where its used. However, it must be noted that these are simply Text Representation Methods—meaning that all four of the methods does not understand the context or meaning behind the words and are simply used to prepare the data for the next step. Further research then, are required to proceed-of which has not been carried out in the study- that captures meaning, semantics and relationships between words specifically Word Embeddings, and Named Entity Recognition.

##### **VI. References**

[1] C. D. Manning and H. Schütze, \*Foundations of Statistical Natural Language Processing\*, Cambridge, MA: MIT Press, 1999.

[2] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proc. North Amer. Chapter Assoc. Comput. Linguistics Human Language Technologies (NAACL-HLT), pp. 4171–4186, 2019.

[3] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," Proc. North Amer. Chapter Assoc. Comput. Linguistics Human Language Technologies (NAACL-HLT), pp. 2227–2237, 2018.

[4] H. Tan and M. Bansal, "LXMERT: Learning Cross-Modality Encoder Representations from Transformers," Proc. 2019 Conf. Empirical Methods Natural Language Processing (EMNLP), pp. 2321–2331, 2019.

‌

‌