Word Embedding

Soft Computing
CSE 4237
Md. Tanvir Rouf Shawon

Word Embedding

- **All texts need to be converted to numbers before starts processing by the machine. Specifically, vectors of numbers.**
- Text is messy in nature and machine learning algorithms prefer well defined fixed-length inputs and outputs.
- **Word Embedding** is one such technique where we can represent the text using vectors. Before deep learning era, the popular forms of word embeddings were:
    a. **BoW**, which stands for Bag of Words
    b. **TF-IDF**, which stands for Term Frequency-Inverse Document Frequency

Bag-of-Words (BoW)

- The **Bag-of-Words (BoW)** model is a way of representing text data when modeling text with machine learning algorithms. The **Bag-of-Words (BoW)** model is popular, simple to understand, and has seen great success in **language modeling** and **document classification**.
- A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:
    - A vocabulary of known words.
    - A measure of the presence of known words.

Bag-of-Words (BoW) - Example

Consider the following 4 sentences:-

- It was the best of times.
- it was the worst of Times.
- it is the time of stupidity.
- it is the age of foolishness.

Form this above example, let's consider each line as a separate **"document"** and the 4 lines as our entire corpus of documents.

Vocabulary

What would be the total vocabulary???

Bag-of-Words (BoW) - Example

1. Design the Vocabulary

The unique words by ignoring case, punctuations, and making them into root words are:

1. it
2. was
3. the
4. best
5. of
6. time
7. worst
8. stupidity
9. is
10. age
11. foolishness

Vocabulary contains 11 words while the full corpus contains 24 words.

Bag-of-Words (BoW) - Example

2. Create Document Vectors

- The objective is to turn each document of text into a vector so that we can use as input or output for a machine learning model.
- Because we know the vocabulary has 11 words, we can use a fixed-length document representation of 11, with one position in the vector to score each word.
- The simplest scoring method is to mark the presence of words as a boolean value, 0 for absent, non-zero (positive value) for present. There can be other methods such as count based methods of the terms if more than one occurrence of a trem.

Bag-of-Words (BoW) - Example

In this example the binary vector of four documents would look as follows:

	it	was	the	best	of	time	worst	stupidity	is	age	foolishness
Document #1 [It was the best of times.]	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**
Document #2 [it was the worst of Times.]	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
Document #3 [it is the time of stupidity.]	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**
Document #4 [it is the age of foolishness.]	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**

BoW

Managing Vocabulary

- In the previous example, the **length of the document vector** is equal to the number of known words which is 11 words.
- For a very large corpus, such as thousands of books, the length of the vector **might be thousands or millions of positions**.
- Further, each document may contain **very few of the known words in the vocabulary**.
- This results in a vector with lots of zero scores, called a sparse vector or sparse representation.
- Sparse vectors require more memory and computational resources **(space and time complexity)**
- It's very important to decrease the size of the vocabulary when using a bag-of-words model.

BoW

Solution #1

There are simple text cleaning techniques that can be used as a first step, such as:

- Ignoring case
- Ignoring punctuation
- Ignoring frequent words that don’t contain much information, called stop words, like “a,” “of,” etc.
- Fixing misspelled words.
- Reducing words to their stem (e.g. “play” from “playing”) using stemming algorithms.

Stemming v/s Lemmatization

Stemming

- Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling.
- For instance, stemming the word ‘Caring‘ would return ‘Car‘.
- Stemming is used in case of large dataset where performance is an issue.

Lemmatization

- Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma.
- For instance, lemmatizing the word ‘Caring‘ would return ‘Care‘.
- Lemmatization is computationally expensive since it involves look-up tables and what not.

BoW

Solution #2

- Each word or token is called a “gram”. Creating a vocabulary of two-word pairs is, in turn, called a **bigram model**.
- An **N-gram** is an N-token sequence of words: a 2-gram (more commonly called a bigram) is a two-word sequence of words like “please turn”, “turn your”, or “your homework”, and a **3-gram (more commonly called a trigram)** is a three-word sequence of words like “please turn your”, or “turn your homework”.
- For example, the bigrams in the first line of text in the previous section: **“It was the best of times”** are as follows:
    - “it was”
    - “was the”
    - “the best”
    - “best of”
    - “of times”
- **A vocabulary then tracks triplets of words is called a trigram model** and the general approach is called the **n-gram model**, where n refers to the number of grouped words.
- **Note: Often a simple bigram approach is better than a 1-gram bag-of-words model.**

One-Hot Representation

- The one hot representation, as the name suggests, starts with a zero vector, and sets as 1 the corresponding entry in the vector if the word is present in the sentence or document.
- Tokenizing the sentences, ignoring punctuation, and treating everything as lowercase, will yield a vocabulary of size 8: {time, fruit, flies, like, a, an, arrow, banana}.
- The binary encoding for **“like a banana”** would then be: [0, 0, 0, 1, 1, 0, 0, 1]

Term Frequency (TF)

Term Frequent (**TF**) is a measure of how frequently a term, t, appears in a document, d:

$$TF_{t,d} = \frac{n_{t,d}}{\text{Total number of terms in document } d}$$

$n_{t,d}$ = Number of times term t appears in a document d. Thus, each document and term would have its own **TF** value.

Consider these 3 documents like **BoW** model:-

- It was the best of the time.
- it was the worst of Times.
- it is the time of stupidity.

TF

The vocabulary or dictionary of the entire corpus would be:-

1. it
2. was
3. the
4. best
5. of
6. time
7. worst
8. is
9. stupidity

Now we will calculate the **TF** values for the **Document 3**.

Document 3 :- **it is the time of stupidity.**

- Number of words in Document 3 = **6**
- TF for the word **'the'** = (number of times **'the'** appears in Document 3) / (number of terms in Document 3) = **1/6**

TF

Likewise:-

- TF('**it**') = 1/6
- TF('**was**') = 0/6 = 0
- TF('**the**') = 1/6
- TF('**best**') = 0/6 = 0
- TF('**of**') = 1/6
- TF('**time**') = 1/6
- TF('**worst**') = 0/6 = 0
- TF('**is**') = 1/6
- TF('**stupidity**') = 1/6

TF

We can calculate all the term frequencies for all the terms of all the documents in this manner:-

Term	Document#1	Document#2	Document#3	TF (Document#1)	TF (Document#2)	TF (Document#3)
it	1	1	1	1/7	1/6	1/6
was	1	1	0	1/7	1/6	0
the	2	1	1	2/7	1/6	1/6
best	1	0	0	1/7	0	0
of	1	1	1	1/7	1/6	1/6
time	1	1	1	1/7	1/6	1/6
worst	0	1	0	0	1/6	0
is	0	0	1	0	0	1/6
stupidity	0	0	1	0	0	1/6

Inverse Document Frequency (IDF)

IDF is a measure of how important a term is. We need the IDF value because computing just the **TF alone is not sufficient** to understand the importance of words:

$$IDF_t = log\,(\frac{\text{Total Number of Documents}}{\text{The Number of Documents with Term } t})$$

IDF

A problem with scoring word frequency is that highly frequent words **('is', 'the', 'a' etc)** start to dominate in the document (e.g. larger score), but may not contain as much **"useful information"** to the model compared to the rarer but **domain specific words**.

One approach is to rescale the frequency of words by **how often they appear in all documents**, so that the scores for frequent words like "the" that are also frequent **across all documents are penalized**.

This approach to scoring is called Term Frequency – Inverse Document Frequency, or TF-IDF for short, where:

- **Term Frequency:** is a scoring of the frequency of the word in the current document.
- **Inverse Document Frequency:** is a scoring of how rare the word is across documents.

Thus the idf of a rare term is high, whereas the idf of a frequent term is likely to be low.

IDF

We can calculate the IDF values for **Document 3**:

Document 3 :- **it is the time of stupidity.**

IDF('it') = log(total number of documents/number of documents containing the word 'it') = log(3/3) = log(1) = 0

We can calculate the IDF values for each word like this. Thus, the IDF values for the entire vocabulary would be:

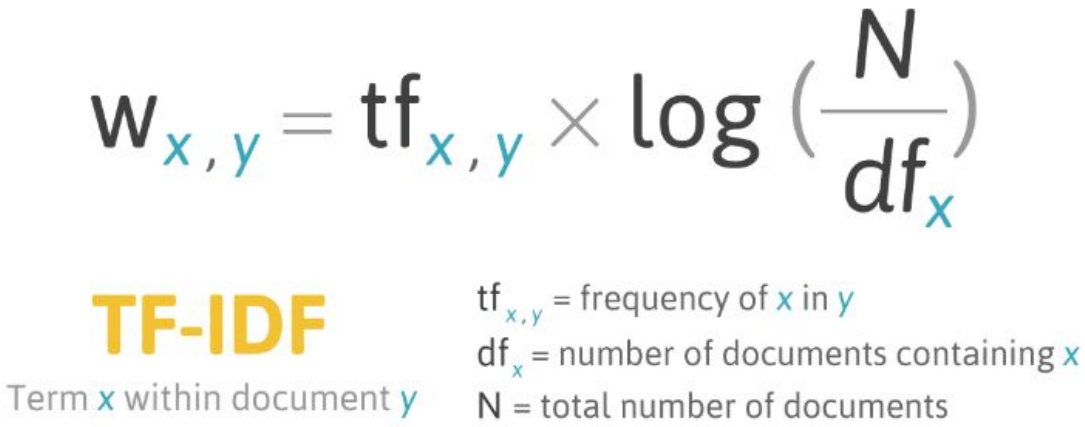
Term	Document#1	Document#2	Document#3	IDF
it	1	1	1	0.00
was	1	1	0	0.18
the	2	1	1	0.00
best	1	0	0	0.48
of	1	1	1	0.00
time	1	1	1	0.00
worst	0	1	0	0.48
is	0	0	1	0.48
stupidity	0	0	1	0.48

TF-IDF

We can now compute the TF-IDF score for each word in the corpus. Words with a higher score are more important, and those with a lower score are less important:

$$(TF - IDF)_{t,d} = TF_{t,d} * IDF_t$$

You can find the overall summary in the following figure.



TF-IDF

We can now calculate the TF-IDF score for every word in **Document 3**:

Document 3 :- **it is the time of stupidity.**

TF-IDF(‘it’, Document 3) = TF(‘it’, Document 3) * IDF(‘it’) = 1/6 * 0 = 0

Likewise:-

- TF(**'it'**) = (1/6) * 0 = 0
- TF(**'is'**) = (1/6) * 0.48
- TF(**'the'**) = (1/6) * 0 = 0
- TF(**'best'**) = (0/6) * 0.48 = 0
- TF(**'time'**) = (1/6) * 0 = 0
- TF(**'of'**) = (1/6) * 0 = 0
- TF(**'stupidity'**) = (1/6) * 0.48

Similarly, we can calculate the TF-IDF scores for all the words with respect to all the documents.

- First, notice how if there is a very common word that occurs in all documents (i.e., n = N), IDF(w) is 0 and the TF-IDF score is 0, thereby completely penalizing that term.
- Second, if a term occurs very rarely, perhaps in only one document, the IDF will be the maximum possible value, log N

Summary

- Bag of Words just creates a set of vectors containing the count of word occurrences in the document, while the TF-IDF model contains information on the more important words and the less important ones as well.
- **Bag of Words vectors are easy to interpret. However, TF-IDF usually performs better in machine learning models.**
- Understanding the context of words is important. Detecting the similarity between the words ‘time’ and ‘age’, or 'stupidity' and 'foolishness'.
- This is where Word Embedding techniques such as **Word2Vec, Continuous Bag of Words (CBOW), Skipgram**, etc come into play.

END