Word Representation

Book Chapter 6

6.1 to 6.5, 6.7 to 6.11.

(Skip 6.6, Backpropagation math of 6.8.2, Equation 6.35 to 6.40)

Semantics: Semantic means meaning

'Dog' & 'Puppy' → different word, similar semantics

River 'Bank' vs Money 'Bank' → Same word, different semantics based on context.

Computer processes symbols (words, tokens) without understanding meaning.

So, our goal is to encode meanings of word or tokens numerically so that it understands.

Representation: How text is converted into numerical form for computer to process.

Main types:

① Sentence / Document level Representation

→ We get one vector that represents entire sentences / documents

② Word level Representation

→ We get one vector for each word

③ Character level representation

→ One vector for each character

Sentence Level Representation

① BoW — Covered in Lecture 3

Problem with BoW:

@ Too sparse:

→ The vocabulary typically are very large. ($|V| = 10\text{-}40k$)

→ Each document has most word missing (0s)

Ex: 10,000 unique word, but, each document has 10 to 100 words. Then, Rest 9990 to 9900 values will be 0 for each vector.

→ Sparsity has other issues like

⇒ high storage cost

⇒ high computation cost

⇒ Easier to overfit.

ⓑ Completely ignore word order:

→ BoW doesn't capture the sequence of the word

Ex: 'dog bite man' & 'man bite dog'

⇒ BoW representation for both of this sentence will be same.

⇒ BoW can't tell that meaning has been reversed because order & relationsheep were lost.

ⓒ Almost no semantic information is preserved

⇒ BoW doesn't know the meaning & relationship.

dog chew snacks
canine eat treats
} totally different representation, similar meaning.

But, BOW works pretty well in simple tasks like spam detection

② TF-IDF:

→ It is a smarter sentence representation.

TF → Term Frequency IDF → Inverse Document frequency.

Idea!

Find words that describe a specific document while ignoring common word that appear every documents.

✱ An informative word/term should:

→ occur many times in some specific context/document (TF)

→ Doesn't occur in every context (IDF)

TF measures importance of a term inside a document. If a word repeats often in a document, then the term is important for that document.

High TF → is locally important & informative

IDF measures uniqueness accross documents. If a word appear in every document (example: is, the, etc) then its uninformative.

High DF → uninformative

High IDF → informative

Combining the idea of TF and IDF we measure the value of TF-IDF for each word/term in document.

$$TF\text{-}IDF = TF * IDF$$

Calculating TF:

$$tf(w,d) = \log\left(1 + f(w,d)\right)$$

- w: word
- d: current document
- $f(w,d)$: how many time word w appear in document d
- $1 +$: prevents $\log(0)$ for missing word
- $\log$ is used to reduce the impact. A word appearing 100 time doesn't mean 100 times more important

Calculating IDF:

$$idf(w, D) = \log\left(\frac{N}{f(w, D)}\right)$$

- w: word
- D: All Document/Corpus
- N: total number of documents
- $f(w, D)$: number of document containing the word w

* The base of log can be 10 or e. Both will work.

Example: [using base e for log], Find tf-idf for d_1, given the dataset:

d_1 = any big cat
d_2 = big cat
d_3 = cat dog cat

here, Number of document, $N = 3$
vocabulary = {any, big, cat, dog}

$tf(any, d_1) = \log(1+1) = 0.693$
$tf(big, d_1) = \log(1+1) = 0.693$
$tf(cat, d_1) = \log(1+1) = 0.693$
$tf(dog, d_1) = \log(1+1) = 0$

$idf(any, D) = \log(3/1) = 1.099$
$idf(big, D) = \log(3/2) = 0.405$
$idf(cat, D) = \log(3/3) = 0$
$idf(dog, D) = \log(3/1) = 1.099$

$tf\text{-}idf(any, d_1) = 0.693 * 1.099 = 0.761$

$tf\text{-}idf(big, d_1) = 0.693 * 0.405 = 0.281$

$tf\text{-}idf(cat, d_1) = 0.693 * 0 = 0$

$tf\text{-}idf(dog, d_1) = 0 * 1.099 = 0$

$\therefore$ Tf idf vector for d_1 = any big cat = $[0.761, 0.281, 0, 0]$

Notice that cat appeared in all documents. Tf-idf assumes that if a word is appearing in every document, then that word is not informative. That is why Idf(cat, D) is zero, so Tf-Idf of cat becomes 0.

Also, the word dog did not appear in first document. So, term frequency (tf) is zero.

Similarly, find tf-idf for d_2 and d_3.

Note that, we don't need to calculate idf again.

Drawback of tf-idf:

1. Still sparse vector for each documents like BoW.
2. No word order. 'dog bites man' & 'man bites dog' has same vector.
3. Almost no semantic information is preserved.

But generally, it works better than BoW

BOW & Tf-idf both gives a vector for a document.

Now, we will dive into word level vectors