NLP
Dense Encodings

Natural Language Processing

Agenda

- The limitation of Sparse Matrices
- Introduction to Dense Encodings
- Different Dense Encoding Models

The limitation of Sparse Matrices

- We have seen **how to represent text data as vectors** using different vectorization techniques such as **BoW** and **TF-IDF** . However, **these techniques often result in sparse vectors**.

- **What is a Sparse Vector ?**

\- A **Sparse vector** is nothing but **a vector having relatively large number of zeros** in it.

For example, the following vector **V** is a Sparse vector -

$$V = [\ 0, 0, 0, 1, 6, 0, 5, 0, 0, 0, 9\ ]$$

Total number of elements = 12
No. of zeros = 8
No. of non-zero elements = 4

The limitation of Sparse Matrices

- **Why is it problematic to have sparse vectors?**

  To understand this, consider the following corpus. By performing text preprocessing (lowercasing, stemming, tokenization, e.t.c) we get the below output:

I love learning.

I enjoy learning NLP.

NLP is the best.

(Corpus)

→ **Preprocessing**

[i, love, learn] ,

[i, enjoy, learn, nlp],

[nlp, best]

- Also, the obtained vocabulary of this corpus is:

I, enjoy, learn, nlp,
love, best

(Vocabulary)

The limitation of Sparse Matrices

- **Why is it problematic to have sparse vectors?**

Now, if we use a vectorization technique, such as BoW, to have a vector representation of the output, we will obtain something like the below, resulting in a sparse matrix:

	i	enjoy	learn	nlp	love	best	
[i, love, learn]	**[1**	**, 0**	**, 1**	**, 0**	**, 1**	**, 0]**	Sparse vector
[i, enjoy, learn, nlp]	**[1**	**, 1**	**, 1**	**, 1**	**, 0**	**, 0]**	
[nlp, best]	**[0**	**, 0**	**, 0**	**, 1**	**, 0**	**, 1]**	Sparse vector

[1, 0, 1, 0, 1, 0]
[1, 1, 1, 1, 0, 0]
[0, 0, 0, 1, 0, 1]

Sparse Matrix

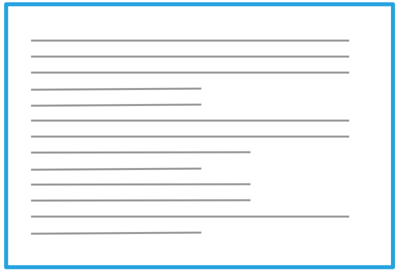
The limitation of Sparse Matrices

- **Why is it problematic to have sparse vectors?**

  If a corpus with 3 sentences can have this many zeros in its vectorized form, we can imagine how sparse the matrix would be for a large amount of data.

- Typically these matrices **contain thousands of zeros**, and also the **dimension of each vector is very high**.

- Training with such data can **result in poor model accuracy** and it is **computationally expensive** as well.



Large amount of data

[1,0,0,0,1,0, …, 0]
[0,0,0,0,1,0, …, 1]
[0,1,0,0,0,0, …, 0]
[1,1,0,0,0,0, …, 0]

vector representation

Introduction to Dense Encodings

- To **deal with the problem of sparse and high dimensional vectors**, **dense encoding** comes into play.

- Dense encoding is **a powerful method** which helps us create **better word representations by reducing dimensionality**.

- The word representation or word embeddings generated by dense encodings are known as '**Dense vectors**'. As the name suggests, these vectors are **compressed** and **contain more relevant information** than sparse vectors.

$$\underbrace{[\,0, 0, 0, 1, 1, 0, 3, 0, 0, 0, 0, 0, 0, \ldots, 1\,]}_{\text{Dimension: 30k+}} \quad \textbf{(Sparse vector)}$$

$$\underbrace{[\,0.5, 3.2, 6.9, 4.5, \ldots, 0.7\,]}_{\text{Dimension: 700+}} \quad \textbf{(Dense vector)}$$

Introduction to Dense Encodings - Dense vectors

- **Dense vectors work better than sparse vectors** in every aspect as every dimension contains relevant information and **they also represent the "semantics"** of text.

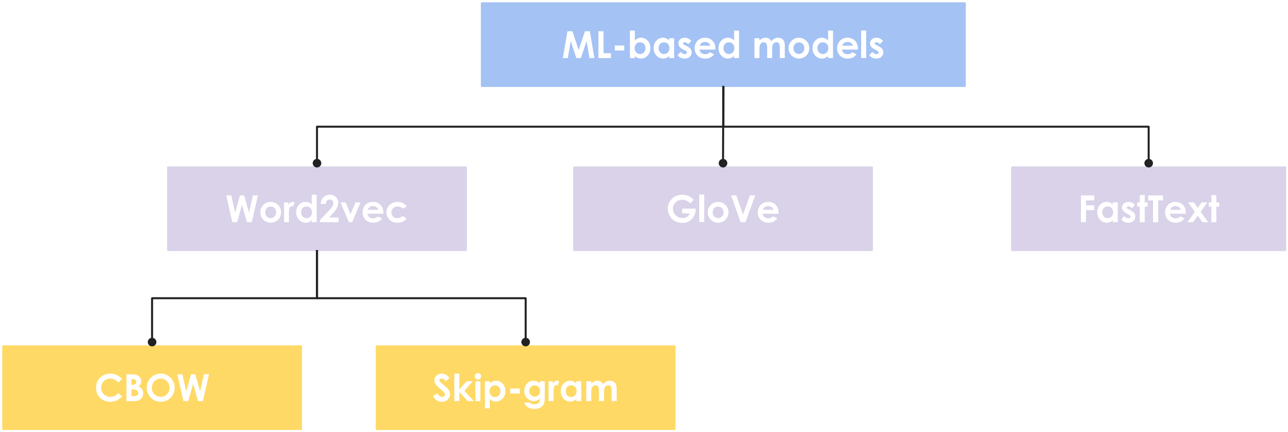
- **What is meant by "Semantics"?**

  Words that occur together in similar contexts often convey meaningful information in some way. For example, the words "**Coffee**", "**Tea**" and "**cold**" may often **occur together when the discussion is about beverages** and, they convey some information. However words like **"Cactus" and "Coffee" would not occur in similar contexts**.

  Dense vectors are able to **capture this information and provide more context** to language, so that models can understand text data with **more accuracy**.

Different Dense Encoding Models

- Some of the Machine Learning models used to generate Dense vectors are:



- Apart from the above ML-based models, there are also advanced **Neural Network based models such as BERT, CoVe**, e.t.c which are **fast**, **efficient to train** and provide us with **high performance**.

Summary

So in order to summarize:

- **Traditional count-based vectorization techniques like BoW, TF-IDF generate sparse vectors** which are difficult to train with due to their high dimensions.

- **Dense encoding helps us deal with this problem by providing vectors with densely packed information.** They have relatively few dimensions in comparison to sparse vectors. These vectors also contain **semantic information** which helps NLP models achieve greater performance.

- Furthermore, there are different types of Machine Learning and Deep learning models to generate dense vectors. We will learn more about them in depth in the next lecture videos.

Happy Learning !