



AIMS

African Institute for
Mathematical Sciences
SENEGAL

Text Classification Using Fully-Connected (FC) NNs

Stephen Kiilu


Ademola Saheed

Mouhamadou Mansour

Ababacar Dioukhane

Overview

1. Introduction
2. Word representation
3. Padding and truncation
4. Methods
5. Results and discussions
6. Conclusion

The background of the slide is composed of two large, overlapping geometric shapes. A teal-colored shape occupies the top-left corner, while a light gray shape occupies the bottom-left corner. The rest of the slide is white. The word "Introduction" is centered in the white area.

Introduction

Introduction and motivation

Why deep learning for text classification?

- ▶ Text classification is one of the popular tasks in natural language processing.
- ▶ The task allows a system to classify a text document into one of the predefined classes.
- ▶ We have seen classical machine learning algorithms like naive Bayes, support vector machine e.t.c can be used for this text, and achieve good results.
- ▶ On the other hand, deep learning methods are proving very good in text classification achieving state-of-the-art results.

Objective of this project

Our primary goal is to implement fully connected (FC) neural networks (NNs) for the task of text classification.

The background of the slide is composed of large, overlapping geometric shapes. A teal-colored triangle is in the top-left corner. A light gray triangle is in the bottom-left corner. The remaining area is white. The text 'Word representation' is centered in the white area.

Word representation

- ▶ Machine learning algorithms take vectors as input
- ▶ When working with text, we need to come up with strategies to vectorize texts.
- ▶ So methods may be inefficient, due to the problem of encoding sparse vectors
- ▶ **Word embedding** is a popular technique used in deep learning, which is a learned representation of words.
- ▶ In word embedding, similar words have similar encoding.
- ▶ This type of representation helps overcome the problem of high-dimensional, sparse vectors

One-hot Vectors

We used **one-hot** vectors to represent words to vectors in the neural network. In **One-hot**, we create a length N vector with all 0s and set the element at position i to 1.

Although this approach is very easy to construct, it is not a good choice because they cannot accurately express the similarities between different words.

id	color
1	red
2	blue
3	green
4	blue



id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

Figure 1: One-Hot Encoding
Source: image by George Novack

Bag of Words (BoW)

- ▶ The simplest word embedding approach

- ▶ document

1	i love maths
2	maths maths is my favourite
3	i love maths and science

- ▶ representation

	and	for	i	is	love	math	my	science
1	0	0	1	0	1	1	0	0
2	0	1	0	1	0	2	1	0
3	1	0	1	0	1	1	0	1

- ▶ It is not a computationally expensive method but it takes a lot of memory. Context information is also lost.

TF-IDF

Is a product of two terms: Term Frequency (TF) and Inverse Document Frequency (IDF).

TF = (Number of Occurences of a word) / (Total words in the document)

IDF = $\text{Log}((\text{Total number of documents}) / (\text{Number of documents containing the word}))$

Pros and cons

It is an improvement over the BoW. However, we still create a huge sparse matrix, and also the semantic relationship of the words is not maintained.

Word2vec

Word2vec is a technique to convert a word into a fixed-length vector by considering the semantic meaning of the words.

The **word2vec** tools contain two models namely skip-gram and CBoW. For Semantically meaningful representations their training relies on conditional probabilities.

Skip-gram

The skip gram model assumes that a word (i.e centre word) can be used to generate its surrounding word in a text sequence.

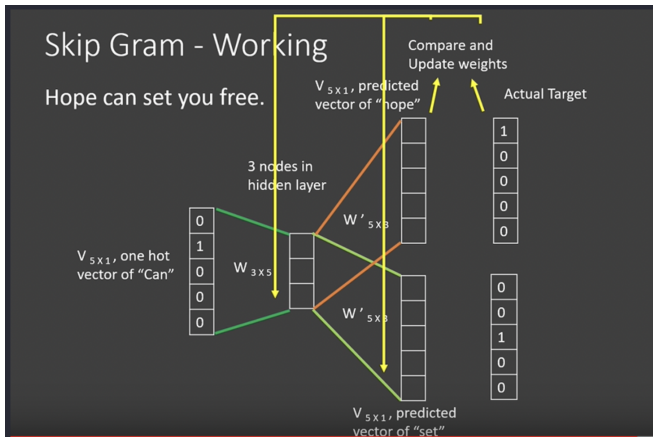


Figure 2: Skip gram Architecture
Source: Suman Kundu-online article

CBOW

The CBOW model is similar to the skip-gram model. The major difference between CBOW to the skip-gram is that CBOW model assumes that a centre word is generated based on its surrounding context words in the text sequence.

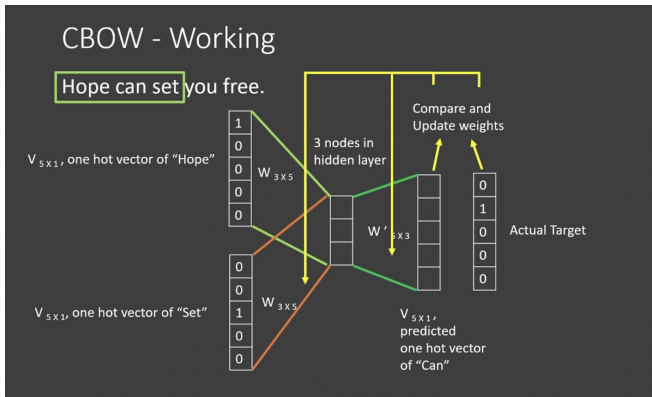


Figure 3: CBOW Architecture
Source: Suman Kundu-online article

Skip-gram model [1]

Each word has two d -dimensional-vector representations. For any word with index i in the dictionary, denote by $\mathbf{v}_i \in \mathbb{R}^d$ and $\mathbf{u}_i \in \mathbb{R}^d$ are center word and a context word, respectively. The conditional probability of generating any context word w_o given the center word w_c can be modeled by a softmax operation on vector dot products:

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}$$

where the vocabulary index set $\mathcal{V} = \{0, 1, \dots, |\mathcal{V}| - 1\}$.

Assume that context words are independently generated given any center word. For context window size m , the likelihood function of the skip-gram model is the probability of generating all context words given any center word:

$$\prod_{t=1}^T \prod_{-m \leq j \leq m, j \neq 0} P(w^{(t+j)} \mid w^{(t)})$$

The skip-gram model parameters are the center word vector and context word vector for each word in the vocabulary. In training, we learn the model parameters by maximizing the likelihood function (i.e., maximum likelihood estimation). This is equivalent to minimizing the following loss function:

$$-\sum_{t=1}^T \sum_{-m \leq j \leq m, j \neq 0} \log P(w^{(t+j)} | w^{(t)})$$

Using stochastic gradient descent to minimize the loss, in each iteration, we can randomly sample a shorter subsequence to calculate the (stochastic) gradient for this subsequence to update the model parameters.

$$\log P(w_o \mid w_c) = \mathbf{u}_o^\top \mathbf{v}_c - \log \left(\sum_{i \in \mathcal{V}} \exp \left(\mathbf{u}_i^\top \mathbf{v}_c \right) \right)$$

Through differentiation, we can obtain its gradient with respect to the center word vector \mathbf{v}_c as:

$$\begin{aligned}\frac{\partial \log P(w_o | w_c)}{\partial \mathbf{v}_c} &= \mathbf{u}_o - \frac{\sum_{j \in \mathcal{V}} \exp(\mathbf{u}_j^\top \mathbf{v}_c) \mathbf{u}_j}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)} \\ &= \mathbf{u}_o - \sum_{j \in \mathcal{V}} \left(\frac{\exp(\mathbf{u}_j^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)} \right) \mathbf{u}_j \\ &= \mathbf{u}_o - \sum_{j \in \mathcal{V}} P(w_j | w_c) \mathbf{u}_j\end{aligned}$$

Note that the gradient calculation requires the conditional probabilities of all words in the dictionary with w_c as the center word.

After training, for any word with index i in the dictionary, we obtain both word vectors \mathbf{v}_i and \mathbf{u}_i . The center word vectors of the skip-gram model are typically used as the word representations.

CBOW Model [1]

For any word with index i in the dictionary, denote by $\mathbf{v}_i \in \mathbb{R}^d$ and $\mathbf{u}_i \in \mathbb{R}^d$. The conditional probability of generating any center word w_c (with index c in the dictionary) given its surrounding context words $w_{v_1}, \dots, w_{v_{\angle m}}$ (with index $v_1, \dots, v_{\angle m}$ in the dictionary) can be modeled by

$$P(w_c \mid w_{o_1}, \dots, w_{o_{2m}}) = \frac{\exp\left(\frac{1}{2m} \mathbf{u}_c^\top (\mathbf{v}_{o_1} + \dots, + \mathbf{v}_{o_{2m}})\right)}{\sum_{i \in \mathcal{V}} \exp\left(\frac{1}{2m} \mathbf{u}_i^\top (\mathbf{v}_{o_1} + \dots, + \mathbf{v}_{o_{2m}})\right)}$$

Let $\mathcal{W}_o = \{w_{o_1}, \dots, w_{o_{2m}}\}$ and $\bar{\mathbf{v}}_o = (\mathbf{v}_{o_1} + \dots, +\mathbf{v}_{o_{2m}}) / (2m)$.

$$P(w_c \mid \mathcal{W}_o) = \frac{\exp(\mathbf{u}_c^\top \bar{\mathbf{v}}_o)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o)}$$

Given a text sequence of length T , where the word at time step t is denoted as $w^{(t)}$. For context window size m , the likelihood function of the continuous bag of words model is the probability of generating all center words given their context words:

$$\prod_{t=1}^T P\left(w^{(t)} \mid w^{(t-m)}, \dots, w^{(t-1)}, w^{(t+1)}, \dots, w^{(t+m)}\right)$$

The maximum likelihood estimation of the continuous bag of words model is equivalent to minimizing the following loss function:

$$-\sum_{t=1}^T \log P \left(w^{(t)} \mid w^{(t-m)}, \dots, w^{(t-1)}, w^{(t+1)}, \dots, w^{(t+m)} \right)$$


Notice that

$$\log P(w_c \mid \mathcal{W}_o) = \mathbf{u}_c^\top \bar{\mathbf{v}}_o - \log \left(\sum_{i \in \mathcal{V}} \exp \left(\mathbf{u}_i^\top \bar{\mathbf{v}}_o \right) \right)$$

Through differentiation, we can obtain its gradient with respect to any context word vector $\mathbf{v}_{o_i} (i = 1, \dots, 2m)$ as

$$\frac{\partial \log P(w_c | \mathcal{W}_o)}{\partial \mathbf{v}_{o_i}} = \frac{1}{2m} \left(\mathbf{u}_c - \sum_{j \in \mathcal{V}} \frac{\exp(\mathbf{u}_j^\top \bar{\mathbf{v}}_o) \mathbf{u}_j}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o)} \right) = \frac{1}{2m} \left(\mathbf{u}_c - \sum_{j \in \mathcal{V}} P(w_j | \mathcal{W}_o) \mathbf{u}_j \right)$$

Continuous bag of words model typically uses context word vectors as the word representations.

The background of the slide is composed of two large, overlapping geometric shapes. A teal-colored shape occupies the upper-left portion, while a light gray shape occupies the lower-left portion. The right side of the slide is a plain white background.

Padding and truncation

Padding

Allows the sequence fed into the neural network to be of the same length by adding 0s. We can perform either **pre** or **post** padding.

Truncation

Tokens of sequence after the 'maximum length' are truncated. We can choose to use either **pre** or **post** truncation.

The Figure 4 below illustrates padding and truncation

sequence before padding

```
[21, 4, 2, 12, 22, 23, 13, 2, 24, 6, 2, 7, 2, 4, 25],  
[ 2, 26, 7, 27, 14, 9, 1, 4, 28 ],  
[15, 25, 1, 29, 6, 15, 30],  
[ 1, 16, 17, 27, 30, 1, 5, 2 ],  
[31, 2, 28, 6, 32, 9, 33],
```



sequence after padding
(padding and truncate in front/pre)

```
[23, 13, 2, 24, 6, 2, 7, 2, 4, 25],  
[ 0, 2, 26, 7, 27, 14, 9, 1, 4, 28],  
[ 0, 0, 0, 15, 25, 1, 29, 6, 15, 30],  
[ 0, 0, 1, 16, 17, 27, 30, 1, 5, 2],  
[ 0, 0, 0, 31, 2, 28, 6, 32, 9, 33],
```

MAX_SEQUENCE_LENGTH = 10

Figure 4: padding and truncation
Source: online-blog

The background of the slide is composed of two large, overlapping geometric shapes. A teal-colored shape occupies the top-left corner, while a light gray shape occupies the bottom-left corner. The rest of the slide is white. The word "Methods" is centered in the white area.


Methods

Model architecture

```
[ ] class FcNeuralNet(nn.Module):
    def __init__(self, input_dim, hidden_dim, num_classes):
        super(FcNeuralNet, self).__init__()
        self.fc1= nn.Linear(input_dim, hidden_dim)
        self.fc2= nn.Linear(hidden_dim, num_classes)

    def forward(self, x):
        """
        The forward pass of the fully connected layer
        """
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        out=F.log_softmax(x, dim =1)
        return out
```

Figure 5: Fully Connected Model Architecture

The background of the slide is composed of large, overlapping geometric shapes. A teal-colored triangle is in the top-left corner. A light gray triangle is in the bottom-left corner. The remaining area is white.

Results and discussions

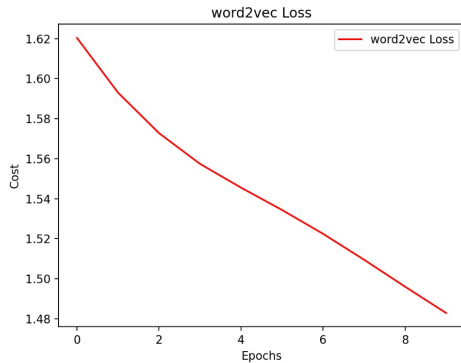
Model Accuracy

The Table 1 below shows the comparison of accuracy (%) results using different word embeddings with different word lengths.

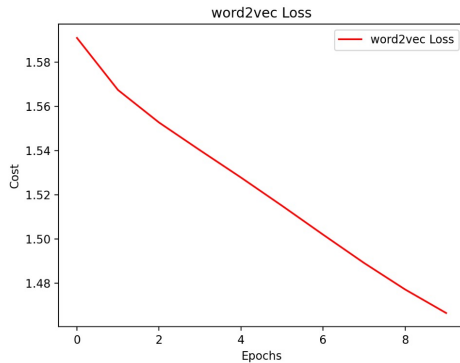
Method	Accuracy
Word2vec w/20 words	35.7
Word2vec w/All words	39.8
GloVe w/20 words	37.9
GloVe w/All words	41.5

Table 1: Results

Model Loss

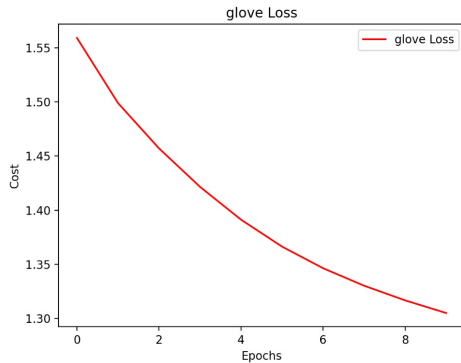


(a) 20 words

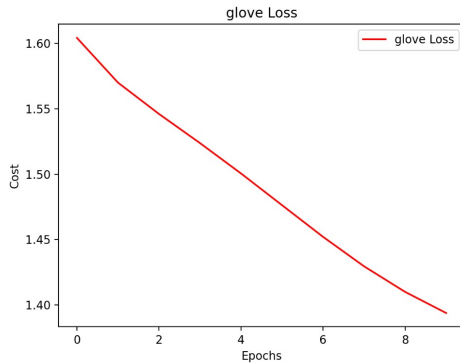


(b) all words

Figure 6: Word2Vec loss



(a) 20 words



(b) all words

Figure 7: GloVe loss

The background of the slide is composed of two large, overlapping geometric shapes. A teal-colored shape occupies the top-left corner, while a light gray shape occupies the bottom-left corner. The rest of the slide is white. The word "Conclusion" is centered in the white area.

Conclusion

Conclusion

- ▶ In this project we trained a fully connected neural network for text classification.
- ▶ We relied on word embeddings, and compared our model for documents of different lengths

Some difficulties

- ▶ The parameters to learn are too many which makes it computationally expensive
- ▶ The accuracy is not good enough

Future Work

- ▶ Work on field based Word2Vec
- ▶ Explore on recurrent neural network RNN or any of its variants (LSTM, GRU).

Dieureudieuf ci dèglou bi !

References I



M. L. Aston Zhang, Zachary C. Lipton and A. J. Smola.

Dive into Deep Learning.

d2l.ai, 0.16.1 edition, 2021.