Sahil Y. Tike
BE B3(212066)

Introduction to Word2Vec

There are two main training algorithms for word2vec, one is the continuous bag of words(CBOW), another is called skip-gram. The major difference between these two methods is that CBOW is using context to predict a target word while skip-gram is using a word to predict a target context. Generally, the skip-gram method can have a better performance compared with CBOW method, for it can capture two semantics for a single word. For instance, it will have two vector representations for Apple, one for the company and another for the fruit.

Gensim Python Library Introduction

Gensim is an open source python

Gensim library will enable us to develop word embeddings by training our own word2vec models on a custom corpus either with CBOW of skip-grams algorithms.

library for natural language processing and it was developed and is maintained by the Czech natural language processing researcher Radim Řehůřek.

In [1]:
```
!pip install --upgrade gensim
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting gensim
  Downloading gensim-4.3.0-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_
64.whl (24.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.1/24.1 MB 586.1 kB/s eta 0:0
0:00m eta 0:00:01[36m0:00:02
Collecting FuzzyTM>=0.4.0
  Downloading FuzzyTM-2.0.5-py3-none-any.whl (29 kB)
Requirement already satisfied: numpy>=1.18.5 in /home/dipali/.local/lib/pytho
n3.8/site-packages (from gensim) (1.24.1)
Requirement already satisfied: smart-open>=1.8.1 in /home/dipali/.local/lib/p
ython3.8/site-packages (from gensim) (6.3.0)
Requirement already satisfied: scipy>=1.7.0 in /home/dipali/.local/lib/python
3.8/site-packages (from gensim) (1.10.1)
Collecting pyfume
  Downloading pyFUME-0.2.25-py3-none-any.whl (67 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.1/67.1 kB 1.0 MB/s eta 0:00:
00 MB/s eta 0:00:01
Requirement already satisfied: pandas in /home/dipali/.local/lib/python3.8/si
te-packages (from FuzzyTM>=0.4.0->gensim) (1.5.3)
Requirement already satisfied: pytz>=2020.1 in /home/dipali/.local/lib/python
3.8/site-packages (from pandas->FuzzyTM>=0.4.0->gensim) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/dipali/.local/
lib/python3.8/site-packages (from pandas->FuzzyTM>=0.4.0->gensim) (2.8.2)
Collecting simpful
  Downloading simpful-2.10.0-py3-none-any.whl (31 kB)
Collecting fst-pso
  Downloading fst-pso-1.8.1.tar.gz (18 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (fr
om python-dateutil>=2.8.1->pandas->FuzzyTM>=0.4.0->gensim) (1.14.0)
Collecting miniful
  Downloading miniful-0.0.6.tar.gz (2.8 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (fr
om simpful->pyfume->FuzzyTM>=0.4.0->gensim) (2.22.0)
Building wheels for collected packages: fst-pso, miniful
  Building wheel for fst-pso (setup.py) ... done
  Created wheel for fst-pso: filename=fst_pso-1.8.1-py3-none-any.whl size=204
30 sha256=9313fbdd5002ccc193a8f4c56cf745f8b17b909dae5d191a52e926142faf5f3f
  Stored in directory: /home/dipali/.cache/pip/wheels/6a/65/c4/d27eeee9ba3fc1
50a0dae150519591103b9e0dbffde3ae77dc
  Building wheel for miniful (setup.py) ... done
  Created wheel for miniful: filename=miniful-0.0.6-py3-none-any.whl size=351
3 sha256=bb9261a76bee4f1a5f9ef4154a2d7d444f5b59adbd3fe49e383f64c009bcba3b
  Stored in directory: /home/dipali/.cache/pip/wheels/ba/d9/a0/ddd93af16d5855
dd9bad417623e70948fdac119d1d34fb17c8
Successfully built fst-pso miniful
Installing collected packages: simpful, miniful, fst-pso, pyfume, FuzzyTM, ge
nsim
Successfully installed FuzzyTM-2.0.5 fst-pso-1.8.1 gensim-4.3.0 miniful-0.0.6
pyfume-0.2.25 simpful-2.10.0
[notice] A new release of pip is available: 23.0 -> 23.0.1
[notice] To update, run: python3 -m pip install --upgrade pip
```

Download the data

Dataset Description

This vehicle dataset includes features such as make, model, year, engine, and other properties of the car. We will use these features to generate the word embeddings for each make model and then compare the similarities between different make model.

In [2]:

```
!wget https://raw.githubusercontent.com/PICT-NLP/BE-NLP-Elective/main/2-Embedd
```

```
--2023-03-01 21:22:40--  https://raw.githubusercontent.com/PICT-NLP/BE-NLP-El
ective/main/2-Embeddings/data.csv (https://raw.githubusercontent.com/PICT-NL
P/BE-NLP-Elective/main/2-Embeddings/data.csv)
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.11
1.133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.1
11.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1475504 (1.4M) [text/plain]
Saving to: ‘data.csv.1’

data.csv.1          100%[===================>]   1.41M   587KB/s    in 2.5s

2023-03-01 21:22:43 (587 KB/s) - ‘data.csv.1’ saved [1475504/1475504]
```

Implementation of Word Embedding with Gensim

In [3]:

```
import pandas as pd
```

In [4]:
```
df = pd.read_csv('data.csv')
df.head()
```

Out[4]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	M
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Tur
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxu
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxu
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	

Data Preprocessing

Since the purpose of this tutorial is to learn how to generate word embeddings using genism library, we will not do the EDA and feature selection for the word2vec model for the sake of simplicity.

Genism word2vec requires that a format of 'list of lists' for training where every document is contained in a list and every list contains lists of tokens of that document. At first, we need to generate a format of 'list of lists' for training the make model word embedding. To be more specific, each make model is contained in a list and every list contains lists of features of that make model.

To achieve this, we need to do the following things

Create a new column for Make Model

In [5]:
```
df['Maker_Model']= df['Make']+ " " + df['Model']
```

Generate a format of ' list of lists' for each Make Model with the following features: Engine Fuel Type, Transmission Type, Driven_Wheels, Market Category, Vehicle Size, Vehicle Style.

In [6]:
```
df1 = df[['Engine Fuel Type','Transmission Type','Driven_Wheels','Market Categ

df2 = df1.apply(lambda x: ','.join(x.astype(str)), axis=1)

df_clean = pd.DataFrame({'clean': df2})

sent = [row.split(',') for row in df_clean['clean']]
```

In [36]:
```
df_clean
```

Out[36]:

	clean
0	premium unleaded (required),MANUAL,rear wheel ...
1	premium unleaded (required),MANUAL,rear wheel ...
2	premium unleaded (required),MANUAL,rear wheel ...
3	premium unleaded (required),MANUAL,rear wheel ...
4	premium unleaded (required),MANUAL,rear wheel ...
...	...
11909	premium unleaded (required),AUTOMATIC,all whee...
11910	premium unleaded (required),AUTOMATIC,all whee...
11911	premium unleaded (required),AUTOMATIC,all whee...
11912	premium unleaded (recommended),AUTOMATIC,all w...
11913	regular unleaded,AUTOMATIC,front wheel drive,L...

11914 rows × 1 columns

Genism word2vec Model Training

We can train the genism word2vec model with our own custom corpus as following:

In [13]:
```
from gensim.models.word2vec import Word2Vec
```

Let's try to understand the hyperparameters of this model.

1. `vector_size` : The number of dimensions of the embeddings and the default is 100.
2. `window` : The maximum distance between a target word and words around the target word. The default window is 5.
3. `min_count` : The minimum count of words to consider when training the model; words with occurrence less than this count will be ignored. The default for min_count is 5.
4. `workers` : The number of partitions during training and the default workers is 3.
5. `sg` : The training algorithm, either CBOW(0) or skip gram(1). The default training algorithm is CBOW.

In [29]:
```
model = Word2Vec(sent, min_count=1,vector_size= 50,workers=3, window =3, sg= 1
```

Save the model

In [30]:
```
model.save("word2vec.model")
```

Load the model

In [31]:
```
model = Word2Vec.load("word2vec.model")
```

After training the word2vec model, we can obtain the word embedding directly from the training model as following.

In [32]:
```
model.wv['Toyota Camry']
```

Out[32]:
```
array([ 0.01411632,  0.14784585,  0.01885239, -0.10753247, -0.07146065,
       -0.22338827,  0.01374025,  0.30203253, -0.09214514, -0.08290682,
        0.04862676,  0.03026489,  0.11046173, -0.02939197, -0.03781607,
        0.17612398,  0.15454124,  0.29968512, -0.13931143, -0.29023492,
       -0.05522037, -0.0564889 ,  0.25777268,  0.07147302,  0.17070875,
        0.00251983, -0.03870121,  0.393018  , -0.04162066, -0.00246239,
        0.01573551,  0.02139783,  0.03586031,  0.00898253,  0.085445  ,
       -0.11928873,  0.1799241 , -0.02834527,  0.04921703,  0.08003329,
        0.10614782, -0.03156348, -0.22044587,  0.09316084,  0.37852806,
        0.0510865 , -0.0260468 , -0.15805483,  0.00056193,  0.01605724],
      dtype=float32)
```

In [33]:
```
sims = model.wv.most_similar('Toyota Camry', topn=10)
sims
```

Out[33]:
```
[('Chevrolet Malibu', 0.9873985648155212),
 ('Nissan Sentra', 0.9869186878204346),
 ('Mazda 6', 0.9854127764701843),
 ('Buick Verano', 0.9837399125099182),
 ('Toyota Avalon', 0.9836648106575012),
 ('Nissan Altima', 0.9834702610969543),
 ('Pontiac Grand Am', 0.9833094477653503),
 ('Chevrolet Cruze', 0.9826680421829224),
 ('Suzuki Verona', 0.9806841611862183),
 ('Suzuki Kizashi', 0.9794840812683105)]
```

Calculate similarity between two words

In [34]:
```
model.wv.similarity('Toyota Camry','Mazda 6')
```

Out[34]:
```
0.98541266
```

In [35]:
```
model.wv.similarity('Dodge Dart','Mazda 6')
```

Out[35]: 0.97065187