Adeeti Kaushal (adeetik2)

Gensim is an open-source library developed by RARE Technologies Ltd and implemented in python by Radim Rehurek. Gensim excels in the Natural Language Processing and is specifically designed do Topic modelling and provides algorithms like LDA (Latent Dirichlet Allocation) and LSI (Latent Semantic Indexing).

According to Gensim official definition
Gensim is a free Python library designed to automatically extract semantic topics from documents, as efficiently (computer-wise) and painlessly (human-wise) as possible. Gensim is designed to process raw, unstructured digital texts ("plain text").

Genism is used in Topic modeling which is used to extract the hidden topics from large volumes of text. Genism internally uses Latent Dirichlet Allocation (LDA) and LSI (Latent Semantic Indexing algorithm for topic modelling.

Core Terminology

- Document: Some string of text.
- Corpus: a collection of documents. It has mainly 2 functions
    - Input for training the model. Gensim focuses on *unsupervised* models so that no human intervention
    - Organize the document and after training, a topic model can be used to extract topics
- Vector: a mathematically representation of a document.
- Model: an algorithm for transforming vectors from one representation to another.

Usages

Description	Code
Downloader API to load datasets	import gensim.downloader as api api.info('glove-wiki-gigaword-50') w2v_model = api.load("glove-wiki-gigaword-50") w2v_model.most_similar('blue')

Create a Dictionary from a list of sentences	import gensim
	from gensim import corpora
	from pprint import pprint
	
	# How to create a dictionary from a list of sentences?
	documents = ["Cancer refers to any one of a large number of diseases development of",
	"abnormal cells that divide uncontrollably and have the ability to infiltrate and destroy",
	"normal body tissue. Cancer often has the ability to spread throughout your body.",
	"Cancer is the second-leading cause of death in the world.",

	"But survival rates are improving", "for many types of cancer", "thanks to improvements in cancer screening and cancer treatment."] texts = [[text for text in doc.split()] for doc in documents] dictionary = corpora.Dictionary(texts) print(dictionary)

Create a Dictionary from one or more text files	from gensim.utils import simple_preprocess
	from smart_open import smart_open
	import os
	
	# Create gensim dictionary form a single tet file
	dictionary = corpora.Dictionary(simple_preprocess(line, deacc=True) for line in open('sample.txt', encoding='utf-8'))
	
	# Token to Id map
	dictionary.token2id

Create a bag of words corpus	import gensim
	from gensim import corpora
	from pprint import pprint
	from gensim.utils import simple_preprocess
	from smart_open import smart_open
	import os
	
	# How to create a dictionary from a list of sentences?
	documents = ["Cancer refers to any one of a large number of diseases development of",
	"abnormal cells that divide uncontrollably and have the ability to infiltrate and destroy",
	"normal body tissue. Cancer often has the ability to spread throughout your body.",
	"Cancer is the second-leading cause of death in the world.",
	"But survival rates are improving", "for many types of cancer",
	"thanks to improvements in cancer screening and cancer treatment."]
	
	texts = [[text for text in doc.split()] for doc in documents]
	dictionary = corpora.Dictionary(texts)
	tokenized_list = [simple_preprocess(doc) for doc in documents]
	# Create the Corpus
	mydict = corpora.Dictionary()
	mycorpus = [mydict.doc2bow(doc, allow_update=True) for doc in tokenized_list]
	pprint(mycorpus)

Create a bag of words corpus from a text file
 	
sample.txt
"The vision of the Digital Container Shipping Association (DCSA) is to shape the digital future of container shipping by being the industry's collective voice, working towards alignment and standardization, setting the frameworks for effective and universally adoptable solutions, exploring possibilities of innovation, and moving the industry forward through standards for IT and non-competitive business practices. from
	from smart_open import smart_open
	import nltk
	from nltk.corpus import stopwords
	import gensim
	from gensim import corpora
	from pprint import pprint
	from gensim.utils import simple_preprocess
	from smart_open import smart_open
	import os
	
	nltk.download('stopwords') # run once
	stop_words = stopwords.words('english')
	
	class BagOfWords(object):
	  def __init__(self, path, dictionary):
	  self.filepath = path
	  self.dictionary = dictionary
	
	  def __iter__(self):
	  global mydict

gensim.utils import simple_preprocess”	for line in smart_open(self.filepath, encoding='latin'):
	# tokenize
	tokenized_list = simple_preprocess(line, deacc=True)
	
	# create bag of words
	bow = self.dictionary.doc2bow(tokenized_list, allow_update=True)
	
	# update the source dictionary (OPTIONAL)
	mydict.merge_with(self.dictionary)
	
	# lazy return the BoW
	yield bow
	
	# Create the Dictionary
	mydict = corpora.Dictionary()
	
	# Create the Corpus
	bow_corpus = BagOfWords('sample.txt', dictionary=mydict) # memory friendly
	
	# Print the token_id and count for each line.
	for line in bow_corpus:
	print(line)

Save and retrieve the dictionary and corpus to disk and load them back	import gensim
	from gensim import corpora
	from pprint import pprint
	from gensim.utils import simple_preprocess
	from smart_open import smart_open
	import os
	
	# How to create a dictionary from a list of sentences?
	documents = ["Cancer refers to any one of a large number of diseases development of",
	"abnormal cells that divide uncontrollably and have the ability to infiltrate and destroy",
	"normal body tissue. Cancer often has the ability to spread throughout your body.",
	"Cancer is the second-leading cause of death in the world.",
	"But survival rates are improving", "for many types of cancer",
	"thanks to improvements in cancer screening and cancer treatment."]
	
	texts = [[text for text in doc.split()] for doc in documents]
	dictionary = corpora.Dictionary(texts)
	# Tokenize the docs
	tokenized_list = [simple_preprocess(doc) for doc in documents]
	# Create the Corpus
	mydict = corpora.Dictionary()
	mycorpus = [mydict.doc2bow(doc, allow_update=True) for doc in tokenized_list]
	#save
	dictionary.save('mydict.dict') # save dict to disk
	corpora.MmCorpus.serialize('bow_corpus.mm', mycorpus) # save corpus to disk
	#reterive
	loaded_dict = corpora.Dictionary.load('mydict.dict')
	pprint(loaded_dict)
	corpus = corpora.MmCorpus('bow_corpus.mm')
	for line in corpus:
	pprint(line)

Create the TFIDF matrix	from gensim import models import numpy as np from gensim import corpora from gensim.utils import simple_preprocess documents = ["Cancer refers to any one of a large number of diseases development of", "abnormal cells that divide uncontrollably and have the ability to infiltrate and destroy",

	"normal body tissue. Cancer often has the ability to spread throughout your body.",
	"Cancer is the second-leading cause of death in the world.",
	"But survival rates are improving", "for many types of cancer",
	"thanks to improvements in cancer screening and cancer treatment."]
	
	# Create the Dictionary and Corpus
	mydict = corpora.Dictionary([simple_preprocess(line) for line in documents])
	corpus = [mydict.doc2bow(simple_preprocess(line)) for line in documents]
	
	# Show the Word Weights in Corpus
	for doc in corpus:
	  print([[mydict[id], freq] for id, freq in doc])
	
	# Create the TF-IDF model
	tfidf = models.TfidfModel(corpus, smartirs='ntc')
	
	# Show the TF-IDF weights
	for doc in tfidf[corpus]:
	  print([[mydict[id], np.around(freq, decimals=2)] for id, freq in doc])

Create bigrams and trigrams	import gensim
	import gensim.downloader as api
	from gensim import models
	import numpy as np
	from gensim import corpora
	from gensim.utils import simple_preprocess
	
	dataset = api.load("text8")
	dataset = [wd for wd in dataset]
	
	dct = corpora.Dictionary(dataset)
	corpus = [dct.doc2bow(line) for line in dataset]
	
	# Build the bigram models
	bigram = gensim.models.phrases.Phrases(dataset, min_count=3, threshold=10)
	
	# Construct bigram
	print(bigram[dataset[0]])
	
	# Build the trigram models
	trigram = gensim.models.phrases.Phrases(bigram[dataset], threshold=10)
	
	# Construct trigram
	print(trigram[bigram[dataset[0]]])

Topic Modeling

- **Using LDA (**Latent Dirichlet Allocation**)**

This is the most popular algorithm for topic modelling. It considers each document as a collection of topics. We need to get the meaning full topics out of the collection in a certain proportion

- Prepare the data: Prepare the data, by removing the stopwords and then lemmatizing it. This can be done by using the pattern package.
- Create Dictionary and Corpus: Use the processed data to create the dictionary.
- Train the data: We need to train the data with topics using the dictionary and corpus created earlier.

- Interpret the LDA output: It mainly consist of
    - Topics in the document
    - What topic each word belongs
    - Phi value: It is the probability of a word to lie in a particular topic

```
from genism.models import LdaModel, LdaMulticore
import genism.downloader as api
from genism.utils import simple_preprocess, lemmatize
from nltk.corpus import stopwords
import re
import logging

stop_words = stopwords.words('english')
stop_words = stop_words + ['com', 'edu', 'subject', 'lines', 'organization', 'would', 'article', 'could']

#load data
dataset = api.load("text8")
data = [d for d in dataset]

#removing stopwords and lemmatize
data_processed = []
for I, doc in enumerate(data[:100]):
   doc_out = []
   for wd in doc:
      if wd not in stop_words:  # remove stopwords
         lemmatized_word = lemmatize(wd, allowed_tags=re.compile('(NN|JJ|RB)'))
         if lemmatized_word:
            doc_out = doc_out + [lemmatized_word[0].split(b'/')[0].decode('utf-8')]
      else:
         continue
   data_processed.append(doc_out)

# Print a small sample
print(data_processed[0][:10])

# Train the
dct = corpora.Dictionary(data_processed)
corpus = [dct.doc2bow(line) for line in data_processed]
lda_model = LdaMulticore(corpus=corpus,
                  id2word=dct,
                  random_state=100,
                  num_topics=7,
                  passes=10,
                  chunksize=1000,
                  batch=False,
                  alpha='asymmetric',
                  decay=0.5,
                  offset=64,
                  eta=None,
                  eval_every=0,
                  iterations=100,
                  gamma_threshold=0.001,
                  per_word_topics=True)

# save the model
lda_model.save('lda_model.model')

# See the topics
lda_model.print_topics(-1)
```

- **Using LSI (**Latent Semantic Indexing**)**
  Modelling is same as LDA, the only difference in training the model.

```
from gensim.models import LsiModel

lsi_model = LsiModel(corpus=corpus, id2word=dct, num_topics=7, decay=0.5)
pprint(lsi_model.print_topics(-1))
```

Conclusion

There are many topic models and word embedding are available in different packages like scikit, R etc. But genism is incomparable with others. It handles large text without handling it in the memory.

References:

https://softwaremill.com/deep-learning-for-nlp/
https://radimrehurek.com/gensim/auto_examples/core/run_core_concepts.html#sphx-glr-auto-examples-core-run-core-concepts-py
http://brandonrose.org/clustering#Latent-Dirichlet-Allocation