GENSIM

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

Gensim is a free Python library designed to automatically extract semantic topics from documents. It is designed to process raw, unstructured digital texts ("plain text"). It is used in Topic modeling which is used to extract the hidden topics from large volumes of text. It internally uses Latent Dirichlet Allocation (LDA) and LSI (Latent Semantic Indexing) algorithm for topic modelling.

Body of Terms:

- Document: String of text.
- Corpus: Collection of documents. It has mainly 2 functions
    - Input for training model. Gensim focuses on unsupervised models so that there will be no human interventions.
    - Organize the document and post training, a topic model can be used to extract topics
- Vector: Mathematical representation of a document.
- Model: An algorithm for transforming vectors from one representation to another.

Usage:

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
	# Create a dictionary from a list of sentences
	documents = ["Diabetes mellitus also called diabetes", "is a term for several conditions involving", "how your body turns food into energy","When you eat a carbohydrate", "your body turns it into a sugar called glucose and sends that to your bloodstream","Your pancreas releases insulin, a hormone that helps move glucose","from your blood into your cells, which use it for energy.","When you have diabetes and don't get treatment", "your body doesn't use

	insulin like it should.","Too much glucose stays in your blood, a condition usually called high blood sugar","This can cause health problems that may be serious or even life-threatening.","There's no cure for diabetes.","But with treatment and lifestyle changes, you can live a long, healthy life.","Diabetes comes in different forms, depending on the cause."] 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 text 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 genism
	from gensim import corpora
	from pprint import pprint
	from gensim.utils import simple_preprocess
	from smart_open import smart_open
	import os
	
	# Create a dictionary from a list of sentences
	documents = ["Diabetes mellitus also called diabetes", "is a term for several conditions involving", "how your body turns food into energy","When you eat a carbohydrate", "your body turns it into a sugar called glucose and sends that to your bloodstream","Your pancreas releases insulin, a hormone that helps move glucose","from your blood into your cells, which use it for energy.","When you have diabetes and don't get treatment", "your body doesn't use insulin like it should.","Too much glucose stays in your blood, a condition usually called high blood sugar","This can cause health problems that may be serious or even life-threatening.","There's no cure for diabetes.","But with treatment and lifestyle changes, you can live a long, healthy life.","Diabetes comes in different forms, depending on the cause."]

	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	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')
	stop_words = stopwords.words('english')
	class BagOfWords(object):
	def __init__(self, path, dictionary):
	self.filepath = path self.dictionary = dictionary def __iter__(self): global mydict
	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 one of the popular algorithms for topic modelling. It considers each document as a collection of topics. We need to get the meaningful topics out of the collection in a certain proportion

  - Data Preparation: 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.
  - Training data: We need to train the data with topics using the dictionary and corpus created earlier.
  - Interpret LDA output: It mainly consist of
    - Topics in the document
    - What topic each word belongs
    - Pi value: It is the probability of a word to be in a particular topic
- Using LSI (Latent Semantic Indexing)

  Modelling is same as LDA, the only difference in training the model.

References:

https://softwaremill.com/deep-learning-for-nlp/

https://radimrehurek.com/gensim/auto_examples/core/run_core_concepts.html#sphx-glrauto-examples-core-run-core-concepts-py

http://brandonrose.org/clustering#Latent-Dirichlet-Allocation