CS 410 - UIUC Text Information Systems (Fall 2021)

TECHNOLOGY REVIEW

LUIS MARIANO OVALLE CASTANEDA

Net ID: lo22

GENSIM

topic modelling for humans

Introduction

GENSIM is an open source Python library for topic modelling, text processing and indexing, it has the capacity of handling large text files since the documents doesn't need to be loaded in totality in memory , it's used primarily for NLP but can also be used as a tool for information retrieval.

Gensim is used by thousands of companies like Amazon, NIM, CISCO, Mindseye, Capital One, etc., for text processing tasks (IR and Mining) like document similarity, recommendation engine, customer complaint exploration (Topic Modelling).

In this Technology Review we will explore the main functionalities and advantages of Gensim, also we will learn how to install it to get started and compare the results of Topic Modeling function to the MP3 labeled data set .

Installation

Gensim runs on most common platforms like Mac OS, Windows and Linux, to install it run the following command from Terminal:

```
pip install --upgrade gensim
```

Another way of installing the library is using **PyCharm** by creating a new file and importing the library:

```
import gensim
```

then hovering over the text select “Install gensim package”, this will download the library and get rid of the unresolved reference.

Note that the Gensim has the following dependencies:

Python 3.6 and up.

Numpy

Smart_open (similar to Python open)

Creating a Dictionary

The first step to use Gensim is to create a **Dictionary**, this structure holds unique terms (**tokens**) from the **Corpus** and an **id number** for each one. A Corpus is a collection of document objects, examples of document objects are a book, a sentence, a tweet, etc. The Corpus will be our BOW (Bag-of-words representation).

Documents can be read from a Python list of strings or from text files, the process of extracting unique terms is called **tokenization,** Gensim provides a function to extract tokens from Documents called **simple_preporcess** (gensim.utils.pre_process) (the following is an example of how to tokenize a text file (we will be using the sample file from MP3):

```
import gensim
import gensim as gensim
from gensim import corpora

processed_list = []

for line in open("test.txt", encoding='utf-8'):
   processed_list.append(gensim.utils.simple_preprocess(line, deacc=True))
```

In the example, we are creating a dictionary with unique terms from the file **test.txt**; each token has a unique id:

```
print(dictionary)

Output:

Dictionary(6 unique tokens: ['mount', 'rainier', 'seattle', 'chicago', 'tower']...)

print(dictionary.token2id)

Output:

{'chicago': 0, 'mount': 1, 'rainier': 2, 'seattle': 3, 'tower': 4, 'willis': 5}
```

For the DBLP.txt file we get similar results, but with a larger dictionary:

```
for line in open("DBLP.txt", encoding='utf-8'):
   processed_list.append(gensim.utils.simple_preprocess(line, deacc=True))

dictionary = corpora.Dictionary(processed_list)

print(dictionary.token2id)

Output (this is the last part of the output):

28719, 'zeros': 28720, 'sdk': 28721, 'hos': 28722, 'reaffirmed': 28723, 'logicbase': 28724,
'psuedo': 28725, 'recursions': 28726, 'amda': 28727, 'externalised': 28728, 'transformable':
28729, 'consulted': 28730, 'clusterfile': 28731, 'obligationswe': 28732, 'mlgf': 28733, 'bison':
28734, 'blender': 28735, 'harmonia': 28736, 'lexer': 28737, 'lexers': 28738, 'lexing': 28739,
'propbank': 28740, 'timeless': 28741, 'scala': 28742, 'toolchains': 28743, 'nahalal': 28744,
'freepastry': 28745, 'ix': 28746, 'kop': 28747, 'arcball': 28748, 'viewcube': 28749,
'underserve': 28750, 'theart': 28751, 'terra': 28752, 'tvmm': 28753, 'seetrieve': 28754,
'subtransactions': 28755, 'racs': 28756, 'reaped': 28757, 'cashmere': 28758, 'grainour':
28759, 'networkboth': 28760, 'treadmarks': 28761, 'letor': 28762, 'listwise': 28763,
'insulates': 28764, 'operationsthere': 28765, 'infant': 28766, 'inqpro': 28767, 'intervene':
28768, 'itss': 28769, 'observedstates': 28770, 'thestructureof': 28771, 'thevariable': 28772,
'uninstantiated': 28773, 'diaspec': 28774, 'wellin': 28775, 'eged': 28776, 'strg': 28777,
'consts': 28778, 'epayment': 28779, 'holiday': 28780, 'overheadwe': 28781, 'statcache':
28782, 'viewwe': 28783, 'ageing': 28784, 'iar': 28785, 'dca': 28786, 'newreno': 28787,
'herself': 28788, 'provokes': 28789, 'obliterating': 28790, 'interferer': 28791, 'interferers':
28792, 'keeper': 28793, 'waterfall': 28794, 'decadesthis': 28795, 'energetic': 28796,
'neutron': 28797, 'λzap': 28798, 'routebricks': 28799, 'terapixel': 28800, 'headache': 28801,
'densitiesthis': 28802, 'tera': 28803, 'usher': 28804, 'chartering': 28805, 'mavcm': 28806,
'refiners': 28807, 'upml': 28808, 'wsmf': 28809, 'motive': 28810, 'compaan': 28811, 'slup':
28812, 'camcorder': 28813, 'imitating': 28814, 'webflow': 28815, 'subsumed': 28816,
'fluent': 28817, 'skepticism': 28818, 'surmounting': 28819, 'pathogen': 28820,
'hypothesised': 28821, 'robber': 28822, 'mymap': 28823, 'parm': 28824, 'callimachus':
28825, 'lions': 28826, 'monterey': 28827, 'toolsthis': 28828, 'cables': 28829, 'cognate':
28830, 'semgram': 28831, 'semgramg': 28832, 'semgramp': 28833, 'tuesday': 28834, 'tvs':
28835, 'dtt': 28836, 'hew': 28837, 'designin': 28838, 'stellation': 28839, 'vsc': 28840,
'emphasised': 28841, 'frost': 28842, 'mentally': 28843, 'taverna': 28844, 'jeopardising':
28845, 'taee': 28846, 'zp': 28847}
```

Creating a Corpus (BOW)

The second structure we will be creating is the Corpus. This object has the term ids and the frequency in each document,

Let’s use the same sample files from MP3 to create our Corpus object:

```
corpus = []
processed_list = []

for line in open("test.txt", encoding='utf-8'):
  processed_list.append(gensim.utils.simple_preprocess(line, deacc=True))

for document in processed_list:
  corpus.append(dictionary.doc2bow(document, allow_update=True))
```

The following is the information of the first ten documents:

```
[(0, 37), (1, 27), (2, 36)]
[(0, 22), (1, 48), (2, 30)]
[(0, 4), (2, 8), (3, 23), (4, 33), (5, 32)]
[(0, 6), (2, 6), (3, 30), (4, 37), (5, 21)]
[(0, 4), (2, 6), (3, 27), (4, 33), (5, 30)]
[(0, 6), (2, 3), (3, 32), (4, 22), (5, 37)]
[(0, 31), (1, 42), (2, 27)]
[(0, 34), (1, 39), (2, 27)]
[(0, 5), (2, 6), (3, 27), (4, 36), (5, 26)]
[(0, 32), (1, 43), (2, 25)]
```

Each line corresponds to a document, for example (0., 37) means that the word with id = 0 appears 37 times in the document, and so on. This is our **term_doc_matrix** from MP3.

Topic Modeling

Now that we have our vocabulary and corpus ready, we can extract the topics using an algorithm like LDA (Latent Dirichlet Allocation), we provide the number of topics, the name of the vocabulary and corpus to the Gensim function (we will keep using our MP3 test data set for this example):

```
model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                        id2word=dictionary,
                                        num_topics=2,
                                        random_state=100,
                                        update_every=1,
                                        chunksize=100,
                                        passes=10,
                                        alpha='auto',
                                        per_word_topics=True)
```

Once we have trained our model, we can check the extracted topics:

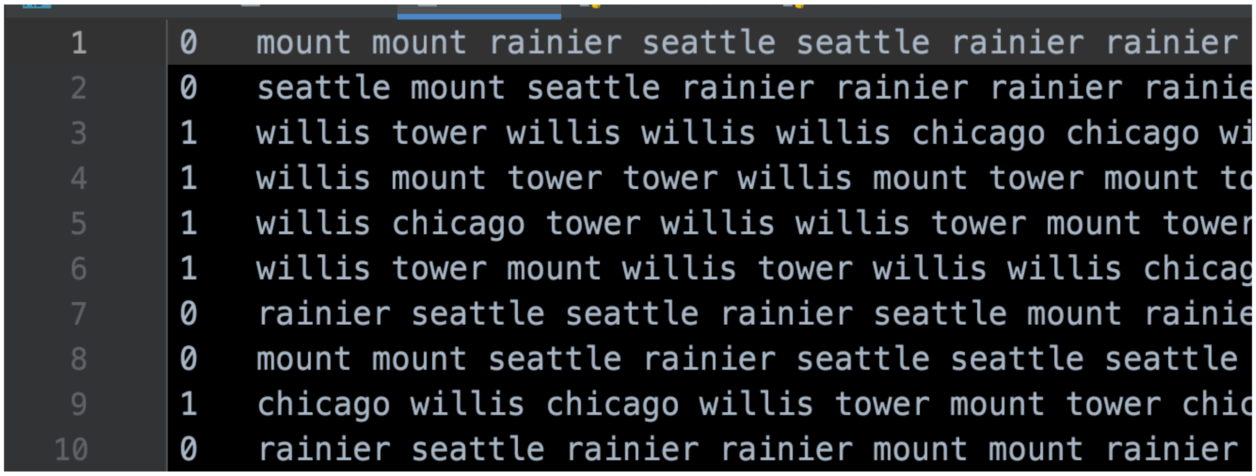
```
for topic in model.show_topics(-1):
   print(topic)

Output:

(0, '0.299*"chicago" + 0.299*"willis" + 0.298*"tower" + 0.053*"mount" + 0.050*"seattle" +
0.001*"rainier"')
(1, '0.396*"rainier" + 0.304*"seattle" + 0.300*"mount" + 0.001*"chicago" + 0.000*"tower"
+ 0.000*"willis"')
```

These are our two topics (**topic_word_prob** matrix), **topic #0 is chicago, and topic #1 is rainier or seattle**. Remember that the sum of probabilities for each topic should add to 1

Now to get the document coverage we can use the following:

```
all_topics = model.get_document_topics(corpus, per_word_topics=True)

index = 0

for doc_topics, word_topics, phi_values in all_topics:
   print("Document ", index, " topics coverage:", doc_topics)
   index = index + 1

Output (first ten documents):

Document  0  topics coverage: [(1, 0.9931711)]
Document  1  topics coverage: [(1, 0.9932823)]
Document  2  topics coverage: [(0, 0.9874911), (1, 0.012508991)]
Document  3  topics coverage: [(0, 0.98766583), (1, 0.0123342285)]
Document  4  topics coverage: [(0, 0.98944205), (1, 0.010557927)]
Document  5  topics coverage: [(0, 0.99021006)]
Document  6  topics coverage: [(1, 0.99324906)]
Document  7  topics coverage: [(1, 0.99323344)]
Document  8  topics coverage: [(0, 0.98865616), (1, 0.011343832)]
Document  9  topics coverage: [(1, 0.99325365)]
```

This is our **document_topic_prob** from MP3, and here we can see to which topic each document belongs. If we compare these results to the labeled data set **test.txt,** we have an exact match. Note **that the topic ids are swapped, chicago = 1 and seattle = 0 in the labeled data set**:



```
0   mount mount rainier seattle seattle rainier rainier
0   seattle mount seattle rainier rainier rainier rainie
1   willis tower willis willis willis chicago chicago wi
1   willis mount tower tower willis mount tower mount to
1   willis chicago tower willis willis tower mount tower
1   willis tower mount willis tower willis willis chicag
0   rainier seattle seattle rainier seattle mount rainie
0   mount mount seattle rainier seattle seattle seattle
1   chicago willis chicago willis tower mount tower chic
0   rainier seattle rainier rainier mount mount rainier
```

Conclusion

In this review, we cover how to install and the basic use of the Gensim toolkit, and compare the results of the Topic Modeling function to the labeled data set provided for MP3 and the results are the same. This toolkit is an excellent alternative to numerous IR and Mining tasks, and knowing the basics should be helpful in future projects.

Reference

Gensim web page

https://radimrehurek.com/gensim/

Selva Prabhakaran: Gensim Tutorial – A Complete Beginners Guide

https://www.machinelearningplus.com/nlp/gensim-tutorial/#9howtousegensimdownloader apitoloaddatasets

Michael Penkov: New Term Topics Methods and Document Coloring

https://github.com/RaRe-Technologies/gensim/blob/develop/docs/notebooks/topic_metho ds.ipynb