# Overview of Gensim [Toolkit]

Gensim (Generate Similar) is a Natural Language Processing Python package for representing raw unstructured digital texts as semantic vectors. It is a free and open source library and licensed under OSI-approved GNU LGPLv2.1 license. For data processing, it uses unsupervised algorithms such as Word2Vec, Latent Semantic Indexing (LSI), Latent Dirichlet Allocation (LDA, LdaModel) etc to find the semantic structures of the documents through statistical analysis of co-occurrence patterns within training documents. Gensim was designed to be as efficient and as human readable as possible. Also it tries to incorporate practicality through commonly used algorithms in industry, scalability through memory independence, and performance through highly optimized algorithm implementation.

The core concepts of Gensim are built around document, corpus, vector and model. A document is considered a text sequence type (e.g., a *str* in Python). Corpus is a collection of documents which can be represented by a *list* of *strings* in Python. If the corpora is very large to load in the memory, Gensim intelligently handles them by streaming one document at a time. Vectors are mathematical representations of documents that are convenient for processing. In Gensim vectors are stored as sparse vectors to save memory and represented as (ID, Value) tuple collection. Here ID can be the ID of a word or a specific query such as how many paragraphs are there in the document. Value can be answers to the queries or frequency of terms. Models are algorithms that transform vectors. They can be trained and saved to disk for transforming new documents later.

## A simple workflow example:

1. Installation
    a. `pip install --upgrade gensim`
2. Loading documents and corpuses
    a. `document = "SyntaxNet: Neural Models of Syntax from Google"`
    b. `text_corpus =["SyntaxNet: Neural Models of Syntax from Google", “Stanfords Name Entity Recognizer (NER)”, “Apache OpenNLP”, “Collaborative filtering algorithms”, "Name Entity Recognition"]`
3. Preprocessing the text - remove stopwords, convert to lower cases etc
    a. `processed_text_corpus = [['syntaxnet:', 'neural', 'models', 'syntax', 'from', 'google'], ['stanfords', 'name', 'entity', 'recognizer', '(ner)'],`

```
['apache', 'opennlp'],
['collaborative', 'filtering', 'algorithms']]
```

4. Building the vocabulary by assigning unique IDs to each term
    a. `from gensim import corpora`
    b. `dictionary = corpora.Dictionary(processed_text_corpus)`
    c. Converting a new document into bag of words format
        i. `dictionary.doc2bow(new_doc.lower().split())`
    d. Converting the original corpus to bag of words vector list,
        i. `bow_corpus = [dictionary.doc2bow(text) for text in processed_text_corpus]`
5. Training a simple tf-idf model to calculate weighting
    a. `from gensim import models`
    b. `new_doc_words = "name syntax".lower().split()`
    c. `tfidf[dictionary.doc2bow(new_doc_words)]`
6. Finding the similarity of a query document
    a. `index = similarities.SparseMatrixSimilarity(tfidf[bow_corpus], num_features=16)`
    b. `query_doc = 'Name Entity Model Validation'.lower().split()`
    c. `query_bow = dictionary.doc2bow(query_doc)`
    d. `similarity_s = index[tfidf[query_bow]]`
    e. `print(list(enumerate(similarity_s)))`
    f. `Output : [(0, 0.0), (1, 0.42153272), (2, 0.0), (3, 0.0), (4, 0.6271355)]` which represents document id and % similarity with the query document

Gensim is designed to handle large collections of documents through Corpus streaming where one document can be handled at a time. Core algorithms are highly optimized and parallelized C methods. Texts are represented here through semantic vectors where models are trained based on these vectors to find semantically related documents. Overall it's an excellent Python library for large scale text information processing.

## References:


[1] Gensim official webpage: https://radimrehurek.com/gensim/index.html
[2] Gensim Tutorial – A Complete Beginners Guide: https://www.machinelearningplus.com/nlp/gensim-tutorial/
[3] Gensim - Introduction: https://www.tutorialspoint.com/gensim/gensim_introduction.htm