A Review of Gensim Toolkit

This paper aims to take a deepdive into the features of gensim toolkit.

Gensim is a free open-source Python library for topic modelling, document indexing and similarity retrieval with large corpora. Gensim can help discover the semantic structure of documents by examining the patterns of words or sentences or entire documents. It uses unsupervised machine learning algorithms to process raw, unstructured texts. This is accomplished by taking a corpus and transforming it into a vector representation of the text in the corpus, which can then be used to train a model. The semantic structure of documents is discovered by examining the statistical co-occurrence patterns within a corpus of training documents. Such a discovery helps in expressing new, semantic relationships and topical similarity between documents.

Gensim makes heavy use of Python's built-in generators and iterators for streamed data processing. The part where Gensim really shines is there is no need for the whole training corpus to reside fully in RAM at any one time and it can process large, web-scale corpora using data streaming. The implementations of vector space algorithms are highly optimized using C, BLAS and memory-mapping.

Basic terminology needed to understand and use gensim

1. Document: raw text.
2. Corpus: a collection of documents.
3. Vector: a mathematically convenient representation of a document.
4. Model: an algorithm for transforming vectors from one representation to another.

Each word in the corpus is associated to a unique integer ID. This is accomplished by using gensim.corpora.Dictionary class. The dictionary we create defines the vocabulary of all the words that are processed.

Gensim implements several popular Vector Space Model algorithms:

1. **TF-IDF (Term Frequency * Inverse Document Frequency)**: The input is a bag-of-words training corpus during initialization The function TfidfModel is used to achieve this transformation

2. **Latent Semantic Indexing LSI or LSA**: The input is either a bag-of-words training corpus or a TfIdf-weighted space. The function LsiModel transforms the former inputs into a latent space of a lower dimensionality.
3. **Random Projections**: This method aims to reduce vector space dimensionality and is a very efficient (both memory- and CPU-friendly) approach to approximating TfIdf distances between documents, by throwing in a little randomness. The function RpModel is used for Random Projections
4. **Latent Dirichlet Allocation LDA**: The function LdaModel is used for LDA transformation. It is a transformation from bag-of-words counts into a topic space of lower dimensionality. LDA is a probabilistic extension of LSA (also called multinomial PCA), so LDA's topics can be interpreted as probability distributions over words. These distributions are, just like with LSA, inferred automatically from a training corpus. Documents are in turn interpreted as a (soft) mixture of these topics (again, just like with LSA).
5. **Hierarchial Dirichlet Process**: HdpModel is a non-parametric bayesian method.

Cosine Similarity, is a standard measure in Vector Space Modeling, widely used in determining the similarity of two vectors. In cases where the vectors are represented by probability distributions different similarity measures like KL-Divergence may be more appropriate.

Some of the most prominent features of Gensim are listed below and a brief overview of their implementations in Gensim are discussed below.

1. Word2Vec Model
2. Doc2Vec Model
3. Ensemble LDA
4. FastText Model
5. LDA Model
6. Word Movers Distance
7. Soft Cosine Measure

Word2Vec Model:

gensim.models.Word2Vec is a model that embeds words in a lower-dimensional vector space using a shallow neural network. The effectiveness of this model comes from its ability to group together vectors of similar words. Given a large enough dataset, Word2Vec can make strong estimates about words meaning based on their occurrences in the text. These estimates yield word associations with other words in the corpus. For example, words like “King” and “Queen” would be very similar with one another. When conducting algebraic operations on word embeddings you can find a close approximation of word similarities.

There are two main architectures of word2vec that are implemented in Gensim.

1. The skip-gram

2. Continous-Bag-of-Words(CBOW)

skip-gram Word2Vec: In the skipgram model architecture word pairs generated by moving a window across text data are taken and a 1-hidden layer neural network is trained to predict the probability of words that are closer to the input word. A virtual one-hot encoding of words goes through a ‘projection layer’ to the hidden layer; these projection weights are later interpreted as the word embeddings.

Continuous-bag-of-words Word2vec is very similar to the skip-gram model. It is also a 1-hidden-layer neural network. The synthetic training task now uses the average of multiple input context words, rather than a single word as in skip-gram, to predict the center word. Again, the projection weights that turn one-hot words into averageable vectors, of the same width as the hidden layer, are interpreted as the word embeddings.

Word2Vec supports several word similarity tasks like get the top ‘n’ most similar words to the input word (function model.wv.most_similar is used to achieve this) and selecting the odd word out of a list of given words(function model.wv.doesnt_match is used to achieve this)

Limitation: one limitation of Word2Vec model is unable to infer vectors for unfamiliar words

Store/Load models: you can store/load models using the standard gensim methods save and load

Hyper Parameters : Parameters that can tuned for training are min_count(for pruning the internal dictionary), vector_size, workers

Training Loss Computation: The parameter compute_loss can be used to toggle computation of loss while training the Word2Vec model. The computed loss is stored in the model attribute running_training_loss and can be retrieved using the function get_latest_training_loss

Doc2Vec Model:

Doc2Vec is a model that represents each Document as a Vector

There are two implementations of the Doc2Vec Model in Gensim:

1. **Paragraph Vector - Distributed Memory (PV-DM)** : This architecture is analogous to Word2Vec Continous-Bag-Of-Words (CBOW)
2. **Paragraph Vector - Distributed Bag of Words (PV-DBOW)** : This architecture is analogous to Word2Vec Skip-gram (SG)

gensim.models.doc2vec.Doc2Vec trains the model with parameters vector_size, min_count, epochs

We can use the trained model to infer a vector for any piece of text by passing a list of words to the model.infer_vector function. This vector can then be compared with other vectors via cosine similarity.

Ensemble LDA Model:

EnsembleLda is a method of finding and generating topics from the results of multiple topic models, it can be used to remove topics from your results that are noise and are not reproducible.

the number of resulting topics varies greatly depending on the clustering parameters which can be provided in the recluster() function or the EnsembleLda constructor. gensim.models.LdaModel is used to train the Ensemble LDA model

Fast Text Model:

Word2Vec does not take into account the morphological structure of a word which carries crucial information regarding the meaning of the word. FastText Model takes this into account and thus

can obtain vectors for words that are not present in the vocabulary. This is achieved by summing vectors for its associated char-ngrams, given that at least one of the char-ngrams be present in the training data. The function gensim.models.fasttext.FastText is used to train the FastText model. Hyperparameters for training the model follow the same pattern as Word2Vec

WordMoversDistance (WMD):

The distance between two documents can be assessed even when there are no words in common by using WMD. It implements Word2Vec[4] vector embedding of words to achieve this. The function wmdistance is used for distance computation, and the WmdSimilarity class for corpus based similarity queries.

Soft Cosine Measure (SCM):

The similarity between two documents can be assessed even when there are no words in common by using SCM. It uses a measure of similarity between words, which can be derived [2] using [word2vec][] [4] vector embeddings of words.

The function inner_product can compute the SCM similarities between two documents .

Conclusions:

Gensim does much more than just Topic Modelling. It is a leading state-of-the-art package for processing texts, working with word vector models (such as Word2Vec, FastText etc) and for building topic models.

It works well with large datasets without having to choke for memory.