Gensim for Topic Modeling: What differentiates it?

Introduction

Topic Modeling refers to the process of creating a probabilistic distribution of “topics” for a given set of documents, also referred to as a “corpus”. Topic modeling is a sub-area of Text Mining, and has lots of potential use cases like, quickly discovering the themes covered by a large corpus of documents like archives of news items, or to get an idea of temporal shift in a research interest of a particular field over years. While there are many sophisticated tools and platforms for NLP, and Text Retrieval and Mining, Gensim is purposely built for tasks which add semantic structure to text, like topic modeling. In this review, I highlight some of the differentiating aspects of Gensim and how it compares to some other tools like Mallet or Stanford Topic Modeling Toolkit.

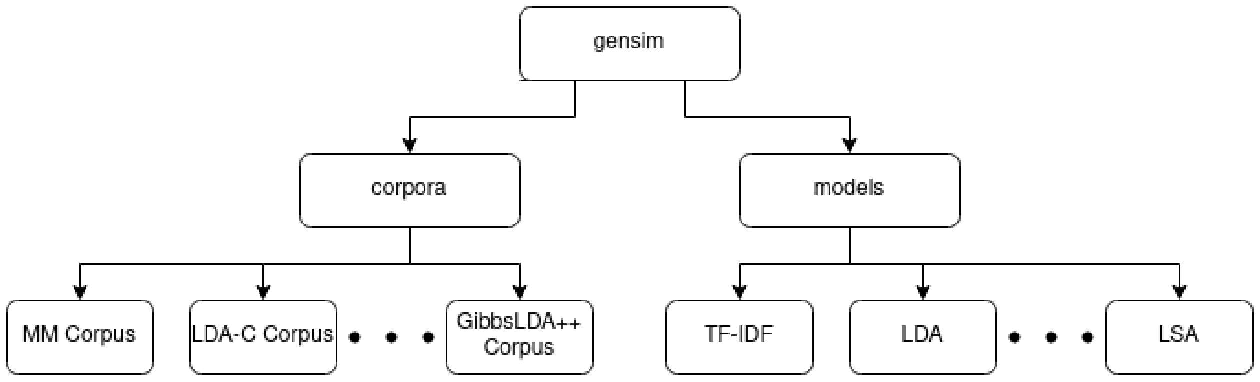
Main Differentiators

I consciously refrained from dwelling into details of how to use Gensim, as such documentation and tutorials are extensively available. Instead, here I focus on some of its core strengths.
In a nutshell, Gensim offers wide range of optimized algorithm implementations (like Word2Vec, FastText, Latent Semantic Indexing (LSI), LDA), developer-friendly APIs and highly memory-efficient programming constructs, which can deal with large web corpuses without need of any vertical scaling. Let's go into a bit of detail below.

Python-based And Intuitive APIs

Gensim is Python based which makes it easier for scores of Data Scientist and engineers to adopt it easily. It can be installed easily using a package manager like “pip” and has a very tiny dependency footprint. As of this writing, it only needs popular packages NumPy and smart_open.

The APIs are cleanly defined and can be broadly classified mainly into two modules: corpora and models.



- The “corpora" module provides APIs to transform a text document into a vector space model (VSM), and broad range of formats in which this VSM can be represented and persisted, like Market-Matrix format (MMCorpus), LDA-C format, GibbsLda++ format etc.

- The interesting thing here is that some of these formats are used in other “popular” implementations of topic modeling (outside Gensim). For example LDA-C format is used by Blei’s LDA C implementation. Similarly, GibbsLda++ format is used by LDA implementation using Gibb’s sampling. So, these corpus representations act as “bridges” through which corpus generated on Gensim can be used on other implementations and vice-versa. This example highlights how Gensim can be used in a data mining pipeline along with other tools.

- Another advantage of using Gensim is the easy integration with the highly popular NLP tool NLTK, which too is Python based. For example, during raw text to corpus creation stage, one can use RegexpTokenizer or WordNetLemmatizer provided by NLTK to tokenize and preprocess the tokens. These tokens can then be seamlessly used by Gensim modules for further processing.

- The “models” module provides apis to transform document vectors from one representation to another. Models are the core engines used to learn Semantic structure from corpus. As mentioned in the introduction section, Gensim provides a wide array of models like Word2vec, LSI, LDA, etc. The API has been designed such that one model can wrap another model. This simplifies the coding by applying model-chaining. For example, the LSI model can wrap the TF-IDF model which is in-turn created using the corpus.

Below table gives a gist of languages and setup details used by various topic modelling tools:

Gensim	**Mallet**	**Stanford Topic Modeling Toolbox**
Python (Comes pre-installed in Mac and Linux)	Java and CLI (Needs JVM in both cases)	Scala (Needs JVM)
Standard installation process using PIP	No standard distribution like Maven central. Manual download and setup needed	Manual download and setup needed
API documentation and lots of tutorials on the official website.	Basic JavaDoc based documentation and limited tutorials on the official website.	No API documentation. Just some basic examples on the official website.

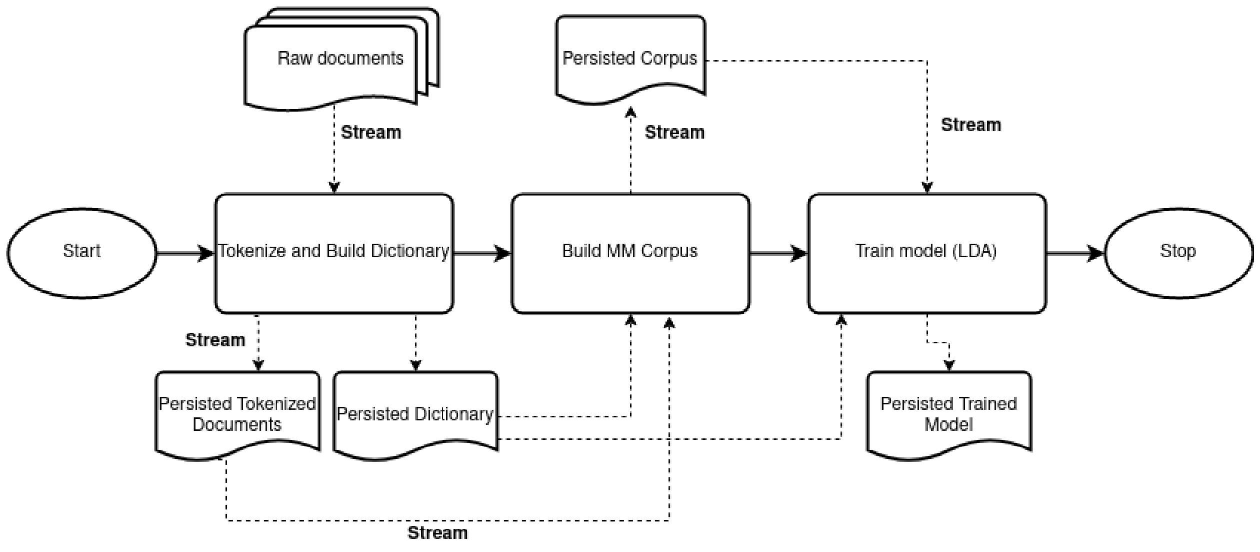
Constant Memory Operation

Any text mining operation on a real-world scale can be memory intensive due to the sheer size of the corpus. Topic modeling is no different. So, if one is not using a commercially licensed software (which is often the case with researchers and students), the task can become difficult to achieve. One of the core tenets of Gensim design is to provide “constructs” for “near” constant memory operations. This is achieved through the use of “streams” and “on-disk” persistence, while building a corpus or training a model.

For example:

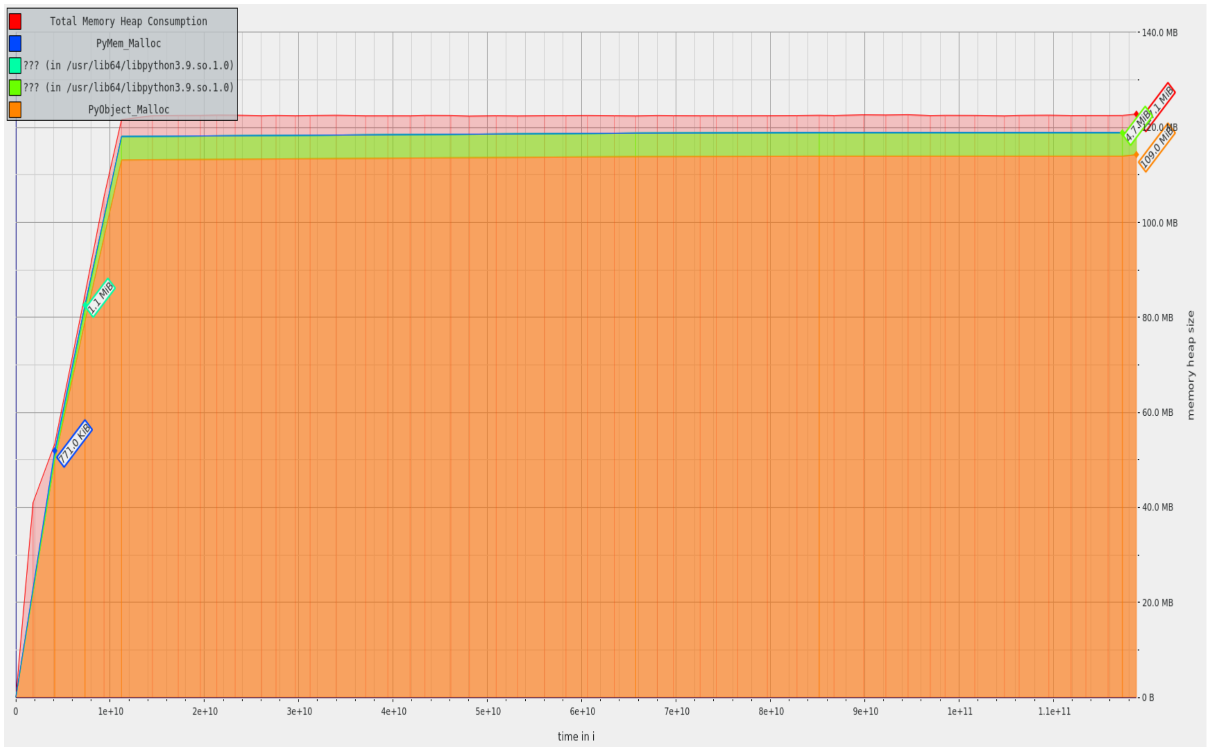
- MM (Market Matrix) corpus API provides support for streamed documents. This means that the process of creating a corpus from raw documents can be done by loading one document at a time in memory.
- Similarly, the LDA model can accept “streamed” corpus instead of loading all corpuses in memory, in one shot.

Below, I depicted a typical “constant” memory workflow for LDA based topic generation:

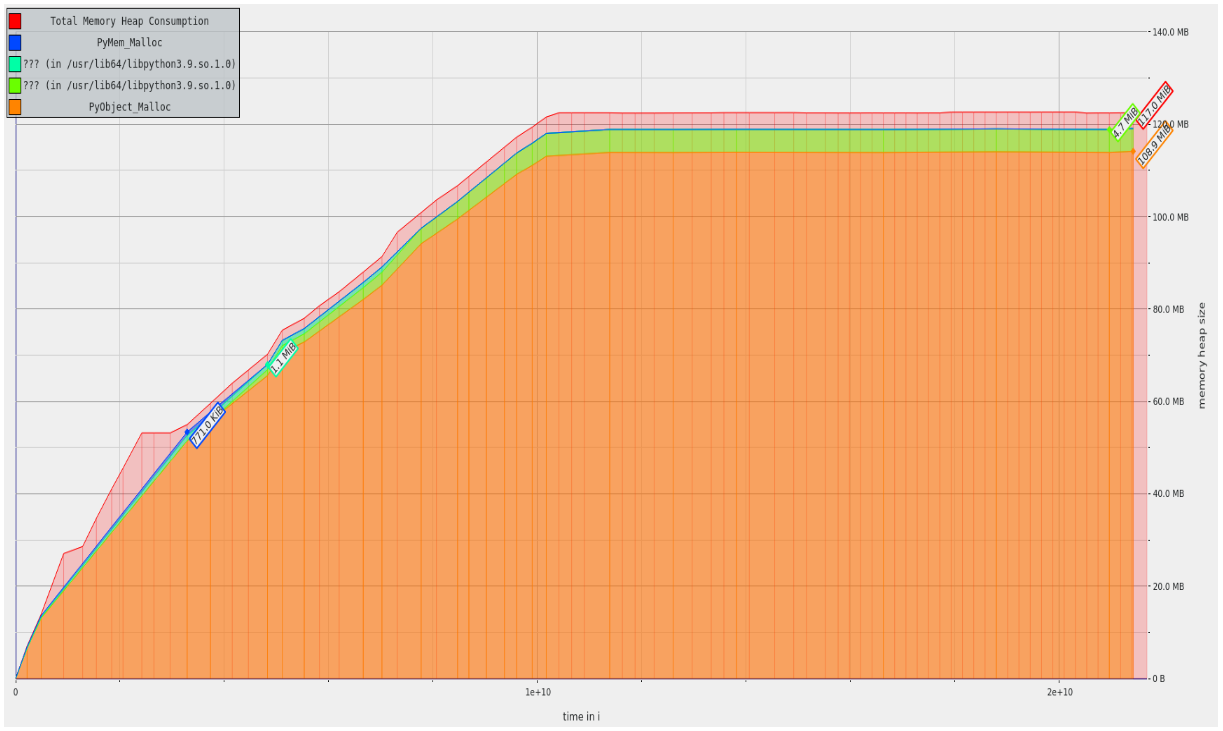


I also did a small experiment to verify the above setup, where I created a corpus with just 5 documents and then with 100 documents, and did a memory profiling for both scenarios, using the Valgrind heap profiler. In both cases, the memory usage was constant ~117 MB, as shown in the diagram below.

Memory Usage for 100 documents corpus



Memory Usage for 5 document corpus



Other tools like Mallet or Stanford Topic Modelling Toolbox do not provide such "constructs" to control the process of building corpus and training model. I used the CLI option of Mallet. It basically requires that Java heap size be adjusted according to the size of the corpus. For example, refer to this tutorial available through the official Mallet website, on how data size related issues should be handled.

Highly Efficient LDA Implementation with Online Training Capability

The LDA (Latent Dirichlet Allocation) implementation in Gensim is based on the LDA variant called 'Online Learning for Latent Dirichlet Allocation' created by Matthew D. Hoffman, David M. Blei, and Francis Bach.

The general approach to estimate the Posterior inference in LDA is by using a batch form of Variational Bayesian (VB). For example, many "popular" LDA implementations use the Gibb's sampling based batch VB algorithm. This requires that the whole corpus to be available locally to create the model. This is not a scalable approach for a huge Corpus like Wikipedia. This algorithm also needs to scan through the whole corpus, whenever a new training document is added.

On the other hand, online learning based LDA, does not need to store or load the whole corpus to build a model. Each document can arrive in a stream and be discarded after one look. This approach has been shown to perform equally well as the batch approach. In addition, this approach makes it possible to do online training, where a new document can be added to the model, without scanning through the whole corpus again.

Gensim's implementation is based on the online learning variation of LDA. This makes it an ideal tool for use cases where we have frequently arriving new (unseen) documents (both for training and testing).

Below is the gist of LDA implementation on the three tools:

Gensim	**Mallet**	**Stanford Topic Modeling Toolbox**
Online Learning LDA	Batch Gibbs Sampling LDA	Batch Gibbs Sampling LDA
Supports streamed corpus	Requires the whole corpus	Requires the whole corpus
Supports distributed (parallel) model training	No support for distributed training	No support for distributed training
Has Implementations of LDA variants like online LDA, Hierarchical LDA, LDA Sequence, as well as LSI model	Has Implementations of LDA like basic LDA, Pachinko Allocation, and Hierarchical LDA	Has implementations of LDA like basic LDA, Labeled LDA and Partial LDA.

Conclusion

Before writing this tech review, I started using Gensim for my Topic modeling project. And I was impressed by the simplicity and intuitiveness of the design. The APIs provide a clean and consistent way to deal with corpus and trained models. With clever use of Python "generators" to stream data and corpus and "local-disk" persistence, one can train models for huge corpuses, with limited memory. Also, the implementation of online learning LDA makes the process more efficient and better suited to dynamic environments like the web. Last but not the least, it is open source software and has good community support. So, based on all the discussed strengths, I would recommend using Gensim for Topic Modeling.

References

https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf
https://github.com/blei-lab/lda-c
http://gibbslda.sourceforge.net
https://www.nltk.org/index.html
https://valgrind.org/docs/manual/ms-manual.html
https://papers.neurips.cc/paper/2010/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf
https://downloads.cs.stanford.edu/nlp/software/tmt/tmt-0.4/
http://mallet.cs.umass.edu/index.php
https://programminghistorian.org/en/lessons/topic-modeling-and-mallet#issues-with-big-data