Hung Nguyen
NETID: hungn2
CS410: Text Information Systems
November 7th, 2021

Tech Review on Gensim

Introduction

The need for robust, reliable, and efficient topic modelling and similarity queries is growing each day. As more and more text and data gets added to databases (mostly via the internet), this need becomes very apparent. There is so much data but also so little time. The obvious solution is to shift from human effort to computer effort. This allows us to mine and extract crucial information from documents to find similarity, and ultimately find relevant documents given a query or determine the overall sentiment of tweets on various topics. There are also many difficulties in using computer effort to find patterns and similarities between documents though: such as efficiency and accuracy/precision issues.

This paper provides a comprehensive review on the Gensim (short for "Generate Similar") toolkit, advertised as "topic modelling for humans." The creator, Radim Rehurek, designed it to maximize efficiency (computer effort), minimize pain (human effort), and also provide accurate and precise results. Best of all, it is free and open-source! Gensim is based on Python and depends mainly on the NumPy and SciPy libraries [2]. The main idea is that users provide a collection of raw text, then *unsupervised* machine learning algorithms are applied to determine the semantic structure of documents. Unsupervised here means that the user does not spend any time on annotations or tagging documents. The more documents are added, the better the model gets at predicting accurate and precise results!

How Gensim Works

Gensim uses vector space algorithms such as Word2Vec, FastText, Term Frequency - Inverse Document Frequency (TF-IDF), Latent Semantic Indexing (LSI), and Latent Dirichlet Allocation (LDA). Given a corpus (collection of documents), it generates vectors for each document under the bag-of-words model to store the frequency of each unique word (token). These vectors can then be transformed via models such as TF-IDF, LSI, and LDA to obtain more useful information.

TF-IDF weighs each token by the frequency of the word throughout the entire corpus (rare words are given a higher weight). This makes sense to do because common words can provide an unnaturally high similarity score between documents even though they're not actually not that similar. Since the tokens are only weighed, the dimensionality of the vector is the same as the unique number of tokens. TF-IDF is useful for finding similarity from querying.

LSI can then be used to reduce the dimensionality of the vector by clustering the words into a specified number of topics. The idea behind LSI is that words close in meaning (and therefore share a similar topic) are assumed to occur in similar documents [3]. For each document, LSI can determine how closely it resembles each of the topics. Thus, LSI is useful to classify documents and group them up in a lower-dimensional space corresponding to various topics. It also helps determine how each word associates with the various topics.

Another neat thing about Gensim is that the documents in the corpus can be streamed in one document at a time. With potentially millions of documents out there, this is very much needed. Due to limitations on RAM, this is very useful because we can feed in one document at a time then return a single document without having to worry about memory issues. As more documents are fed in, the LSI model also gets trained and becomes more representative of the given corpus. Naturally, as more documents are in the corpus, the model will become better at predicting the topics and similarity of future documents.

LDA is similar to LSI in that it also reduces the dimensionality of the vectors into a specified number of topics. It extends LSI by probability by defining topics as probability distribution over words. And like LSI, the probability distributions change as more documents are added to the training corpus.

Another great thing about LSI and LDA is that words given in a query doesn't necessarily need to be *present* in relevant documents. Because the model is trained to determine what words are similar, it will be able to find relevant documents that don't contain the words in the query, yet are still relevant to the query. This happens more frequently than we think, so LSI and LDA is a really good way to make sure we find similar documents even if they don't share the exact common words.

Adopters of Gensim

Due to the power and efficiency of gensim, many companies out in industry have adopted the use of Gensim in their products. Some of the noteworthy, big name companies that are using Gensim are the following [2]:

1) Amazon for document similarity.
2) Cisco Security for large-scale fraud detection.
3) Capital One for topic modeling customer complaints and exploration.
4) Tailwind for posting relevant and interesting content to Pinterest.
5) Talentpair to match candidates to relevant jobs.

For my class project, I am planning on querying for topics on the UIUC Slack Channel and Campuswire to quickly find relevant entries in both platforms given a user query. I have no doubt that Gensim will be very useful here and make my life easier.

Conclusion

Gensim is a toolkit that truly simplifies similarity querying and topic modeling. Using the powerful capabilities of NumPy, which has proven its efficiency in large matrix operations, along with proven vector space models like TF-IDF, LSI, and LDA, Gensim generates useful data for similarity querying and topic modelling. As the amount of text data grows each day (mainly via the internet), the need for topic modelling using machines to help us understand the vast amount of information also grows. Gensim provides a solution to satisfy this need. It continues to get better each day as more training data is added.

References:

1. Gensim HomePage
   <https://radimrehurek.com/gensim/index.html>

2. Gensim Github Documentation
   <https://github.com/RARE-Technologies/gensim/>

3. Latent Semantic Analysis
   <https://en.wikipedia.org/wiki/Latent_semantic_analysis>