Anchored CorEx: Hierarchical Topic Modeling with Minimal Domain Knowledge

Topic modeling by way of **Cor**relation **Ex**planation (CorEx) yields rich topics that are maximally informative about a set of data. This project optimizes the CorEx framework for sparse binary data, allowing for topic modeling over large corpora. In addition, this code supports hierarchical topic modeling, and provides a mechanism for integrating domain knowledge via anchor words and the information bottleneck. This semi-supervised anchoring is flexible and allows the user to anchor words through creative strategies that promote topic representation, separability, and aspects.

Unlike LDA, the CorEx topic model and its hierarchical and semi-supervised extensions make no assumptions on how documents are generated and, yet, they still find coherent, meaningful topics as measured across a variety of metrics. Our TACL paper makes detailed comparisons to unsupervised and semi-supervised variants of LDA:

Gallagher, Ryan J., Kyle Reing, David Kale, and Greg Ver Steeg. "Anchored Correlation Explanation: Topic Modeling with Minimal Domain Knowledge." *Transactions of the Association for Computational Linguistics (TACL)*, 2017.

This code can be used for any sparse binary dataset. In principle, continuous values in the range zero to one can also be used as inputs but the effect of this is not well tested.

Getting Started

Install

Python code for the CorEx topic model can be installed via pip:

```
pip install corextopic
```

Running the CorEx Topic Model

Given a doc-word matrix, the CorEx topic model is easy to train. The code follows the scikit-learn fit/transform conventions.

```
import numpy as np
import scipy.sparse as ss
from corextopic import corextopic as ct

# Define a matrix where rows are samples (docs) and columns are features (words)
X = np.array([[0,0,0,1,1],
              [1,1,1,0,0],
```

```
                [1,1,1,1,1]], dtype=int)
# Sparse matrices are also supported
X = ss.csr_matrix(X)
# Word labels for each column can be provided to the model
words = ['dog', 'cat', 'fish', 'apple', 'orange']
# Document labels for each row can be provided
docs = ['fruit doc', 'animal doc', 'mixed doc']

# Train the CorEx topic model
topic_model = ct.Corex(n_hidden=2)  # Define the number of latent (hidden) topics to use.
topic_model.fit(X, words=words, docs=docs)
```

Once the model is trained, the topics can be accessed through the `get_topics()` function.

```
topics = topic_model.get_topics()
for topic_n,topic in enumerate(topics):
    words,mis = zip(*topic)
    topic_str = str(topic_n+1)+': '+','.join(words)
    print(topic_str)
```

Similarly, the most probable documents for each topic can be accessed through the `get_top_docs()` function.

```
top_docs = topic_model.get_top_docs()
for topic_n, topic_docs in enumerate(top_docs):
    docs,probs = zip(*topic_docs)
    topic_str = str(topic_n+1)+': '+','.join(docs)
    print(topic_str)
```

Summary files and visualizations can be outputted from `vis_topic.py`.

```
from corextopic import vis_topic as vt
vt.vis_rep(topic_model, column_label=words, prefix='topic-model-example')
```

The visualizations utilize `seaborn`, and `graphviz` is used for plotting hierarchical topic models. Graphviz should be compiled with the triangulation library for the best visual results.

Full details on how to retrieve and interpret output from the CorEx topic model are given in the example notebook.

Hierarchical Topic Modeling

Building a Hierarchical Topic Model

For the CorEx topic model, topics are latent factors that can be expressed or not in each document. We can use these binary topic expressions as input for another layer of the CorEx topic model, yielding a hierarchical representation.

```
# Train the first layer
topic_model = ct.Corex(n_hidden=100)
topic_model.fit(X)

# Train successive layers
tm_layer2 = ct.Corex(n_hidden=10)
tm_layer2.fit(topic_model.labels)

tm_layer3 = ct.Corex(n_hidden=1)
tm_layer3.fit(tm_layer2.labels)
```

Visualizations of the hierarchical topic model can be accessed through `vis_topic.py`.

```
vt.vis_hierarchy([topic_model, tm_layer2, tm_layer3], column_label=words, max_edges=300, prefix
```

Choosing the Number of Topics

There is a principled way for choosing the number of topics within each layer of the topic model. Each topic explains a certain portion of the *total correlation* (TC). These topic TCs can be accessed through the `tcs` attribute, and the overall TC (the sum of the topic TCs) can be accessed through `tc`. To assess how many topics to choose at each layer, you may look at the distribution of `tcs` for each layer. As a rule of thumb, additional latent topics should be added until additional topics contribute little to the overall TC.

To get better topic results, you can restart the CorEx topic model several times from different initializations, and choose the topic model that has the highest TC (explains the most information about the documents).

Semi-Supervised Topic Modeling

Using Anchor Words

Anchored CorEx allows a user to anchor words to topics in a semi-supervised fashion to uncover otherwise elusive topics. If `words` is initialized, anchoring is straightforward:

```
topic_model.fit(X, words=words, anchors=[['dog','cat'], 'apple'], anchor_strength=2)
```

This anchors "dog" and "cat" to the first topic, and "apple" to the second topic. As a rule of thumb `anchor_strength` should always be set above 1, where setting `anchor_strength` between 1 and 3 gently nudges a topic towards the anchor words, and setting it above 5 more strongly encourages the topic towards the anchor words. We encourage users to experiment with `anchor_strength` for their own purposes.

If `words` is not initialized, you may anchor by specifying the integer column feature indices that you wish to anchor on. For example,

```
topic_model.fit(X, anchors=[[0, 2], 1], anchor_strength=2)
```

anchors the features of columns 0 and 2 to the first topic, and feature 1 to the second topic.

Anchoring Strategies

In our TACL paper, we explore several anchoring strategies:

1. *Anchoring a single set of words to a single topic*. This can help promote a topic that did not naturally emerge when running an unsupervised instance of the CorEx topic model. For example, one might anchor words like "snow," "cold," and "avalanche" to a topic if one suspects there should be a snow avalanche topic within a set of disaster relief articles.

```
topic_model.fit(X, words=words, anchors=[['snow', 'cold', 'avalanche']], anchor_strength=4)
```

2. *Anchoring single sets of words to multiple topics*. This can help find different aspects of a topic that may be discussed in several different contexts. For example, one might anchor "protest" to three topics and "riot" to three other topics to understand different framings that arise from tweets about political protests.

```
topic_model.fit(X, words=words, anchors=['protest', 'protest', 'protest', 'riot', 'riot', 'riot
```

3. *Anchoring different sets of words to multiple topics.* This can help enforce topic separability if there appear to be chimera topics. For example, one might anchor "mountain," "Bernese," and "dog" to one topic and "mountain," "rocky," and "colorado" to another topic to help separate topics that merge discussion of Bernese Mountain Dogs and the Rocky Mountains.

```
topic_model.fit(X, words=words, anchors=[['bernese', 'mountain', 'dog'], ['mountain', 'rocky',
```

The example notebook details other examples of using anchored CorEx. We encourage domain experts to experiment with other anchoring strategies that suit their needs.

Note, when running unsupervised CorEx, the topics are returned and sorted according to how much total correlation they each explain. When running anchored CorEx, the topics are not sorted by total correlation, and the first *n* topics will correspond to the *n* anchored topics in the order given by the model input.

Technical notes

Binarization of Documents

For speed reasons, this version of the CorEx topic model works only on binary data and produces binary latent factors. Despite this limitation, our work demonstrates CorEx produces coherent topics that are as good as or better than those produced by LDA for short to medium length documents. However, you may wish to consider additional preprocessing for working with longer documents. We have several strategies for handling text data.

0. Naive binarization. This will be good for documents of similar length and especially short- to medium-length documents.

1. Average binary bag of words. We split documents into chunks, compute the binary bag of words for each documents and then average. This implicitly weights all documents equally.

2. All binary bag of words. Split documents into chunks and consider each chunk as its own binary bag of words documents. This changes the number of documents so it may take some work to match the ids back, if desired. Implicitly, this will weight longer documents more heavily. Generally this seems like the most theoretically justified method. Ideally, you could aggregate the latent factors over sub-documents to get 'counts' of latent factors at the higher layers.

3. Fractional counts. This converts counts into a fraction of the background rate, with 1 as the max. Short documents tend to stay binary and words in long documents are weighted according to their frequency with respect to background in the corpus. This seems to work Ok on tests. It requires no preprocessing of count data and it uses the full range of possible inputs. However, this approach is not very rigorous or well tested.

For the python API, for 1 and 2, you can use the functions in `vis_topic` to process data or do the same yourself. Naive binarization is specified through the python api with count='binarize' and fractional counts with count='fraction'. While fractional counts may be work theoretically, their usage in the CorEx topic model has not be adequately tested.

Single Membership of Words in Topics

Also for speed reasons, the CorEx topic model enforces single membership of words in topics. If a user anchors a word to multiple topics, the single membership can be overriden. Going forward, we plan to develop a multi-membership extension of the CorEx topic model that retains the computational efficiency.

Underlying Theory and Motivation of CorEx

Discovering Structure in High-Dimensional Data Through Correlation Explanation, Ver Steeg and Galstyan, NIPS 2014.
Maximally Informative Hierarchical Representions of High-Dimensional Data, Ver Steeg and Galstyan, AISTATS 2015.