EVALUATING TOPIC MODEL DIMENSIONALITY REDUCTION PERFORMANCE

NICHOLAS LINES

This paper provides added details to complement the talk, available on YouTube. The slides, code, and all files needed to reproduce the experiment are available on GitHub at https://github.com/linesn/topic_model_dimensionality_reduction. Please refer to the talk for illustrations and diagrams, and refer to the code for implementation information and numerical results.

1. Introduction

Probabilistic Topic Models have become increasingly popular since their birth in the early 2000s, because they enable readers to approach large text corpora using themes and document similarity information to sift to just the content of interest. An important aspect of the topic model problem is dimensionality reduction. However, the bulk of work toward evaluating topic model quality seems to neglect the dimensionality reduction aspect of these tools. In this experiment we'll demonstrate a practical approach to measure both the topic quality and the dimensionality reduction quality of two popular algorithms in a realistic problem setting.

2. Topic Modeling

Consider a situation in which you encounter a large collection of text documents which you must sort through and review in order to find the content you are interested in. A common approach in Natural Language Processing (NLP) is to begin by dropping from each document all ordering and syntax, preserving only the counts of vocabulary words observed, thereby forming a "bag-of-words-model" of the documents. By representing each document as a row vector whose entries are counts of vocabulary words in that document, we form the $n \times v$ Document-Word matrix, where n is the number of documents in the corpus and v the number of words in the working vocabulary. Typically the vocabulary is reduced significantly both for performance and computational purposes. This may include the removal of stopwords (low-information common words), combining multiple words into a single token via stemming and lemmatization, and other tokenization practices.

The topic modeling problem is as follows: given a Document-Word matrix, and a stated number of topics t, factor the matrix as

$$\overset{n\times v}{\text{Document-Word}} \approx \overset{n\times t}{\text{(Document-Topic)}}\ \overset{t\times v}{\text{(Topic-Word)}}$$

such that

Date: November 16, 2021.

(1) the pairwise document relationships in the Document-Topic Embedding are appropriately similar to their analogs in the Document-Word embedding, and
(2) the Topic-Word matrix is sufficiently coherent.

We are intentionally vague in describing the optimization conditions, since these vary greatly in practice.

By far the most popular approach to topic modeling is Latent Dirichlet Allocation (LDA) [2]. This Bayesian technique assumes that latent Dirichlet priors exist from which the topic-word and document-topic distributions are sampled. LDA recovers the parameters to these priors. Typically the step of inferring topics uses Gibb's Sampling, a powerful Markov Chain Monte-Carlo technique.

Another popular option for topic modeling is simple Non-negative Matrix Factorization. This approach is particularly appropriate for factoring the large and sparse Document-Word count matrix. NMF minimizes the squared reconstruction error

$$\|\text{Document-Word } - (\text{Document-Topic})\ (\text{Topic-Word})\|^2$$

plus some regularization terms. In the above expression, the norm is typically the Frobenius norm. Optimization is usually done by alternatively improving each factor matrix.

NMF offers some distinct advantages compared to LDA. It is quite likely to be faster, since it does not require the lengthy serial steps of topic inference, and could be easily parallelized. However, LDA has a reputation for delivering higher quality topics. Topic quality is often measured in coherence. In our case, we will use David Mimno's U_mass coherence, because this will allow us to investigate the quality of individual topics and average the topic coherences to judge the quality of the entire model. We write the document frequency (i.e. count of documents including it) of a word v as $D(v)$, and the co-document-frequency (i.e. count of documents including both) of two words u, v as $D(u, v)$. Let $V^{(\tau)} = \left(v_1^{(\tau)}, \ldots, v_M^{(\tau)}\right)$ be the list of the M most frequently occurring words in topic τ. The U_mass coherence is defined as

$$C\left(\tau; V^{(\tau)}\right) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{D\left(v_m^{(\tau)}, v_l^{(\tau)}\right) + 1}{D\left(v_l^{(\tau)}\right)},$$

and it tells us the essentially whether frequently occurring words in this topic cooccurr frequently, a feature in good, distinct topic models. Bigger values are better. Typically users choose $M = 20$ as the number of top words to examine.

These are not the only contenders in the topic model arena, however. A recent addition to this list is Top2Vec [1], which takes a very different approach from traditional topic modeling. Top2Vec begins by representing documents in a semantic embedding space, and then performs dimensionality reduction with Uniform Manifold Approximation and Projection (UMAP) [5] and clustering with Hierarchical Density Based clustering (HDBSCAN) [4]. UMAP is a dimensionality reduction approach designed to faithfully reproduce local topological structure in high dimensional data projections to lower dimensions, particularly to viewable dimensions. Top2Vec fills many of the gaps presented by traditional topic modeling algorithms like LDA or NMF. For example, it removes the need for vocabulary reduction, provides a clear approach for topic (cluster) number choice, and allows for easy

hierarchical expansion and compression of models. On the other hand, Top2Vec is limited to hard clustering, while LDA and NMF traditional approaches allow fuzzy clustering.

3. Evaluating Dimensionality Reduction Algorithms

In this section we'll introduce a few tools for measuring the quality of dimensionality reduction. A more thorough description can be found in [8]. Most dimensionality reduction evaluation techniques begin by comparing the pairwise distances between data points in the two embeddings. Let $\boldsymbol{x} = x_1, x_2, ..., x_n$ be a set of n points in one embedding E, and let $\boldsymbol{x'} = x'_1, x'_2, ..., x'_n$ be their analogs embedded into E' by a dimensionality reduction algorithm A. The pairwise distance matrices in the two embeddings are given by

$$D = \{D_{i,j} = \|x_i - x_j\|_d\}, D' = \{D'_{i,j} = \left\|x'_i - x'_j\right\|_{d'}\},$$

where typically the distance functions $\|\cdot\|_d$ and $\|\cdot\|_{d'}$ are equivalent and chosen to be the Euclidean distance between the points.

Computing these distances can be computationally intensive when n is large and when the dimension of E is high. We also do not typically care about each distance to the same degree: dimensionality reduction is usually considered successful if nearby points retain relative distances and are stay roughly far away from further neighbors. We don't really care if clusters are made more distant, provided the points within the clusters stay close to each other. This motivates us to switch to thinking in terms of the ranking matrix

$$R \text{ such that } R_{i,j} = \#\left\{k : D_{i,k} < D_{i,j} \text{ or } (D_{i,k} = D_{i,j} \text{ and } k < j)\right\},$$

where # denotes the size or cardinality of the set, i.e. the number of events matching the description. The ranking matrix ranks which points are the most and least distant from each other point. We can similarly define R' in the new embedding. These two matrices ignore most of the information contained in the distance matrices and instead focus on the structure of the data's graphs.

From the two ranking matrices we can form the coranking matrix for the algorithm A,

$$Q \text{ such that } Q_{i,j} = \#\left\{(k,l) : R_{k,l} = i \text{ and } R'_{k,l} = j\right\}.$$

Thus each element of Q tells us how many times A reordered the ith nearest point to a reference point as the jth nearest point. This matrix has useful properties: The bottom right corner represents changes to the far-off element orderings, which are trivial. The top left corner represents changes to nearby orderings, which are serious. We'll call the upper right half of the matrix *extrusions,* incidents where neighboring points are moved further apart in rank. The lower left half represent *intrusions,* incidents where neighboring points are moved closer in rank. Since we only really care about the upper left hand corner of the coranking matrix, for values of k between 1 and n, we define the Q-nearest-neighbors function

$$Q_{NN}(k) = \frac{1}{kn} \sum_{(i,j):i \leq k, j \leq k} Q_{i,j},$$

which is simply the normalized sum of the values in that upper left-hand $k \times k$ square. Note that the normalization multiplier $1/k$ grows larger when smaller k-neighborhoods are chosen. By taking the area under this curve

$$AUC_{Q_{NN}} = \frac{1}{n} \sum_{k=1}^{n} Q_{NN}(k)$$

we obtain a single real-valued number that represents how well A preserves the local k-neighborhood structure of the data in its mapping. A value of 1 is perfect, and corresponds to a mapping that changes no ranks. A value of 0 corresponds to a random mapping.

Many other tools like this have been devised, some focusing more on extrusions or more on intrusions, and others focusing only on certain k-neighborhoods. For our experiment, however, the area under the Q_{NN} curve will serve as our only metric for dimensionality reduction quality.

We won't use reconstruction error, because the LDA tool that we'll use in this experiment will perfectly reconstruct the Document-Word matrix, by design. We will track the NMF reconstruction error, nonetheless.

4. Experiment Details

For our experiment we used the first 2000 articles in the English CNN Daily Mail dataset [3], which is easily obtained through HuggingFace or other NLP data repositories. Data preprocessing is not a significant part of this experiment, so we did not optimize this step. We processed the data using Scikit-Learn's TF-IDF vectorizer and pyStemmer to drop vocabulary words that appear fewer than 5 times total or more than at least once in 50% of the documents[1]. This gave us a sparse-matrix object containing the Document-Word matrix X that will be used for the experiment. We then used Dask's multiprocessing library and mlflow's model metrics tools to train vanilla LDA, NMF, and UMAP dimensionality reductions for $t = 2, 3, ..., 200$ as the choice for the number of topics. For each model at each run we compute[2] the local k-neighborhood structure preservation $AUC_{Q_{NN}}$, the mean coherence $\frac{1}{t}\sum_{i=1}^{t} C\left(\tau_i; V^{(\tau_i)}\right)$, the normalized reconstruction error $\|X - (\text{Document-Topic})(\text{Topic-Word})\|^2 / \|X\|^2$, and the training time.

We originally hypothesized that NMF would see higher $AUC_{Q_{NN}}$ values than LDA but lower than UMAP, indicating middling performance in neighborhood structure preservation, and that LDA would have by far the highest coherence values.

In the interests of reproducibility, each random process was seeded. We also tracked Python package management in Conda with an environment file that can be used to recreate the software environment.

5. Results

We discovered that the mean coherence for NMF and LDA did not become distinguishable until after 25 to 50 topics are used, far more than the 9 clusters

[1]This process is recommended by Johnathan Soma in [7].

[2]We will use a fast Python implementation of this function provided by Tim Sainburg in [6].

that a PCA would suggest[3] is optimal. Even for topic numbers where LDA received a much higher mean coherence score, LDA witnessed lower worst-case individual topic scores than NMF did. This suggests that NMF performed as well as LDA for lower topic number choices, and provided more stability though inferior quality in higher dimensions.

NMF saw slightly lower $AUC_{Q_{NN}}$ scores than UMAP for the first few topic numbers (2,3, etc.) but quickly surpassed UMAP, continuing to grow long after UMAP plateaued at around 10 topics. LDA, by contrast, remained little better than random in its preservation of local neighborhood structures. This suggests that NMF is producing much higher quality Document-Topic matrices than LDA, and that UMAP can't compete with it in higher dimensions than the visible choices. NMF's reconstruction errors grow gradually as higher topic numbers are chosen, but in the more likely region (around 9), they are comparable to LDA's perfect reconstruction.

Finally, NMF and UMAP are consistently fast (˜30 seconds max) while LDA climes linearly in run-time with t, reaching as high as 3300 seconds for a single run.

Our main takeaway is that for small choices (fewer than 50) of number of topics, NMF is a superior approach, since it produces better quality dimensionality reduction and equivalent quality topics, and in much less time than LDA. Even in higher dimensions, one should be aware that LDA trades speed and neighborhood structure integrity for higher-quality topics.

6. Future Work

Several clear future tasks are highlighted by this experiment. First, we may wish to rerun this experiment using different metrics for the quality of each factor matrix. We also should confirm these results in different languages and document types, to ensure that the results are generalizable.

More importantly, though, these results have suggested that NMF outperforms UMAP in terms of preserving local k-neighborhood structures, while keeping a comparable run-time. This implies that Top2Vec might perform better with NMF as it's dimensionality reduction step in place of UMAP. This would make for a great future experiment.

7. Conclusion

Topic models are becoming more popular, and with that popularity comes a need for justification and evaluation. The reader has hopefully been convinced by this work that the Document-Topic matrix is not a trivial element in the topic modeling problem. Topic model evaluation should take into account both the quality of the Document-Topic matrix and the Topic-Word matrix, allowing the model designer to adjust the prioritization of either matrix to determine the choice of modeling tool and parameters used.

References

1. Dimo Angelov, *Top2vec: Distributed representations of topics*, (2020).

[3]PCA is often used to guess the right number of clusters or topics in a problem like this. In this case we ran PCA over the matrix X and found the elbow in the explained variance curve by applying a 8th degree smoothing polynomial, and finding the first zer of its second derivative.

2. David M Blei, Andrew Y Ng, and Michael I Jordan, *Latent dirichlet allocation*, the Journal of machine Learning research **3** (2003), 993–1022.
3. Karl Moritz Hermann, TomÃąs KociskÃ¡, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom, *Teaching machines to read and comprehend*, NIPS, 2015, pp. 1693–1701.
4. Leland McInnes, John Healy, and Steve Astels, *hdbscan: Hierarchical density based clustering*, Journal of Open Source Software **2** (2017), no. 11, 205.
5. Leland McInnes, John Healy, and James Melville, *Umap: Uniform manifold approximation and projection for dimension reduction*, arXiv preprint arXiv:1802.03426 (2018).
6. Tim Sainburg, *Fast computation of a coranking matrix for dimensionality reduction with python, joblib, and numba.*
7. Johnathan Soma, *Choosing the right number of topics for scikit-learn topic modeling: Data science for journalism.*
8. Yinsheng Zhang, Qian Shang, and Guoming Zhang, *pydrmetrics-a python toolkit for dimensionality reduction quality assessment*, Heliyon **7** (2021), no. 2, e06199.

E-mail address: nicholasalines@gmail.com