Algorithm Overview

Ensemble LDA

Table of Contents

1. How the Algorithm Works.....1
2. Example.....4
3. Pseudocode.....7

1. How the Algorithm Works

Assume topic A and B to be true topics that are generated by an algorithm. Topic AB is the mixture of both of those topics, to which the classic model converges sometimes.

$$AB = (A + B) / 2$$

To properly detect topic mixtures, topic A looks at topic AB and sees that **there is** a large portion of itself inside that mixture topic. Vice versa, when topic AB looks at topic A, it will see that there is a large chunk of itself **missing** inside topic A. By doing so, topic A can be detected as a **core** by the DBSCAN algorithm, whereas this is not the case for AB. And for this to work, one has to use an asymmetric distance matrix for the DBSCAN algorithm, which causes the distance from A to AB to be lower than the reverse distance.

The process of looking at how much of topic A is represented in topic B – which is used as the distance from A to B – , can be done by:

1. Keep as many from the most likely words from topic A, so that the probability of them sums up to 95%. This means the algorithm selects the largest 95% of the words by mass. Remove the other words, which will throw away noise and unimportant words. Create a binary mask in that fashion with a 0 for removed words and a 1 for important words in that topic.
2. Remove the words from topic B that correspond to a zero in the mask of step 1, to create the filtered topic B. Do the same to topic A. Both topics A and topic B are now filtered based on the mask that is extracted from topic A. Note, that this might temporarily throw away words in topic B, that have a high probability. The goal is to see how good topic A is represented in topic B, so the algorithm doesn’t care about the other words.

3. Calculate the cosine distance between the filtered topic B and the filtered topic A. Since all probabilities are >= 0, the maximum distance is 1. If the sum of probabilities in the filtered topic B is smaller than 5%, it is either mostly orthogonal to A or empty (which is represented by noise and/or uniform distributions over words) and the distance is set to the maximum of 1.

$$\delta(A,B) = 1 - \frac{A \cdot B}{|A| \cdot |B|}$$

4. Write that distance down into the pairwise asymmetric distance matrix. It is an asymmetric distance matrix, because the distance in one direction (A → C, mask is created from A) is different than the distance in the other (C → A, mask is created from C). Because of the masking in the ensemble LDA, the distance from the filtered topics A to B is different than the return distance. When topic A is represented in a topic mixture AB, that topic mixture AB is not well represented in topic A.
5. Repeat this process for every possible pair of topics of all the trained models from the ensemble. This includes pairs that are made out of topics from two different models, as well as topics from the same model.
6. Now apply DBSCAN to cluster the topics based on the distances from the pairwise distance matrix. When the distance from node A to B is **small**, node **A** is a valid topic. When the return distance, that means from node B to node A, is **large** at the same time, **B** is not valid. In that case B still supports A in becoming a core, but won't become a core of that cluster itself. For a topic to be identified as a core, multiple topics ("neighbors") need to be close by. In the python package, this threshold defaults to half of the number of models.
7. Similar to DBSCAN, the neighbors need to check their distances to their neighbors as well to become an extension of the cluster. However, they also need to have a distance smaller or equal to epsilon to at least 25% of the cores that are already in the cluster. This makes sure that the cluster maintains a compact size instead of growing inifinitely. We call this algorithm check-back DBSCAN (CBDBSCAN).

   As the clustering takes a negligible amount of time to finish compared to the training of individual models. the results can easily be reclustered with different hyper parameters to get the best result possible.

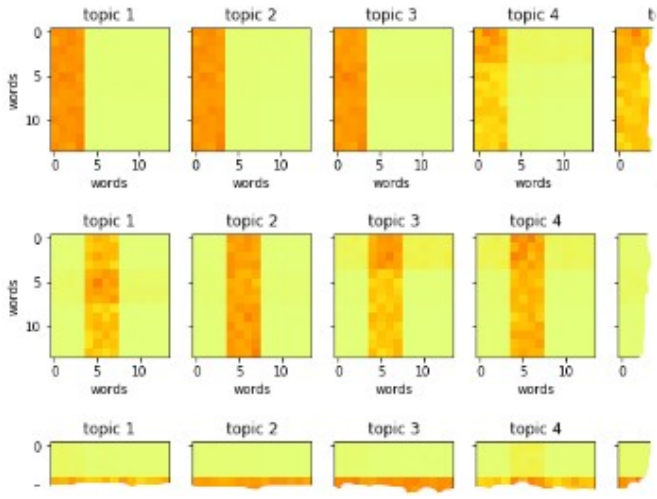

Figure 1: This is an example of how similar topics from various models will be clustered together. The mean of the topics in each cluster will represent a stable topic.

8. When a cluster is large enough, that means when it contains enough cores, it will be identified as topic. In the python package this minimum threshold is currently the number of models divided by 4 plus 1. Also, the minimum threshold is not higher than 3 cores and always 1 core or more.
9. The last step is to take the mean of the topic word distributions of each core and use this as the topic that represents the cluster.

For a more visual example, a corpus of 3 different words and documents that are a mixture of those words can be considered. The documents are mostly influenced by only one of the three words, but sometimes the other two words are also apparent, which makes the documents appear noisy in Figure 2.

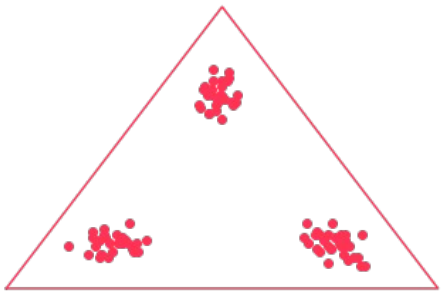

Figure 2: The documents placed in a word simplex.

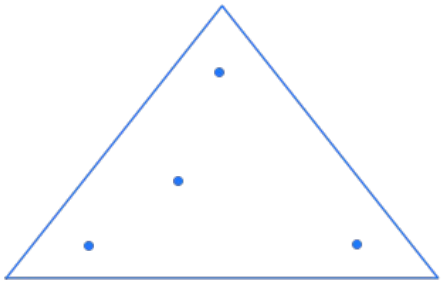

Figure 3: Now an LdaModel is being trained on that corpus with num_topics set to 4. Originally, 3 clusters were visible so obviously this parameter is wrong. However, the number of underlying topics is not known as it is an unsupervised algorithm. The fourth topic appears as topic mixture, which is the point that is close to the middle.

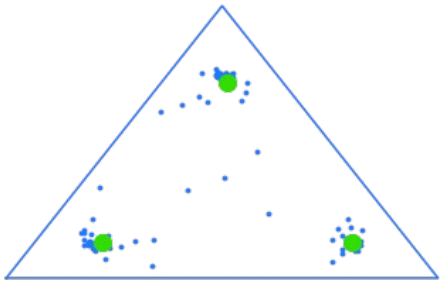

Figure 4: After training many models, the results can be clustered to remove topic mixtures. The green points are the results of Ensemble LDA.

2. Example

1. Assume the following example topics. T_k is the distribution of words in a topic.

T_1:

House 0.6	Garden 0.3	Roof 0.06	Floor 0.04

T_2:

House 0.6	Garden 0.2	Roof 0.19	Floor 0.01

2. Sort and set **mask** indices on T_1 based on the most characteristic 95% of the words in T_1 by mass, which will get rid of the noise, to calculate mask(T_1, T_2). So the probabilities are summed up from left to right as long as the sum is lower than 0.95.

T_1:

House 0.6	Garden 0.3	Roof 0.06	Floor 0.04

T_1:

1	1	0	0

3. **Mask** T_2. Masking means to select elements from the distribution T_2 based on the binary numbers from the list above. The total probability does not need to sum up to 1 after removing an element, because the cosine distance does not care about the magnitude.

T_2:

House 0.6	Garden 0.3

4. Now do the same for mask(T_1, T_1), which just throws away the lower 5% of the distribution by Mass. As can be seen here, mask(T_1,T_1) is very similar to mask(T_1,T_2):

T_1:

House 0.6	Garden 0.2

5. now the **cosine distance** has to be calculated. This is done using the following formula, which uses the dot product between vectors:

$$M_{12} = \delta(T_{1,}T_2) = 1 - \frac{mask(T_{1,}T_1) \cdot mask(T_{1,}T_2)}{|mask(T_{1,}T_1)| \cdot |mask(T_{1,}T_2)|}$$

$$M_{12} = 0.01$$

All those M_{ij} distances for each pair form the distance matrix M. The distance is very small in this example, which means the two topics T_1 and T_2 are very close together.

6. After doing this for a few topics, the result might be like this. 0.01 was rounded to 0 here:

M =

	1	2	3	…
1		**0**	0.3	…
2	0		0.2	…
3	0.9	0.2		…
…	…	…	…	…

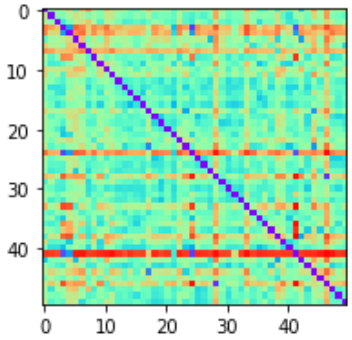

Figure 5: Example of a real world distance matrix. Cold is low, warm is high distance.

It is asymmetric because the pair ij does not result into the same as ji, it might be very similar or very dissimilar. The return distance to a distance is the number in this matrix on the opposite side of the diagonal.

7. It will **cluster** based on the distance between those topics using **CBDBSCAN**. Now assume Epsilon is 0.5. So Topic 1 (T_1) is close enough to T_2 and T_3. Assuming the required number of neighbors for a core is 2, it will successfully become a core of the first cluster. This is the first core of that cluster, so the used “Check Back” step is not required.

8. The algorithm will continue to search for members of that cluster recursively and proceed to T_2. The distance **from** T_2 **to** T_1 and T_3 is small as well, so T_3 will be the second core for the cluster, but only if the distance to 25% of the existing members of the cluster is also small. In this example it is obviously true, as it is close to the single existing member of the cluster.

9. The next element to check for is T_3, which is only close enough to T_2. Note, that the distance to T_1 is large, even though the distance from T_1 to T_3 is small. This might be the case because T_3 is a mixture of multiple topics. Since it does not exceed the number of required neighbors, it will not be identified as a core.

10. If the cluster has more elements than a predefined threshold, the mean of the cores is used in order to generate a new stable topic