**Workflow:**

1. Prepare texts for the LDA model training (“with all regulations present in EUR-Lex”)

You can use “2\_step…” notebook I’ve shared with you as a reference.

* 1. Get texts of regulations
  2. Clean and standardize each text:
     1. Delete stop-words
     2. Lemmatize
     3. Select only meaningful lemmas (based on POS distribution) most likely nouns, proper nouns, verbs, and adjectives (keep the order of words in each sentence during transformation for future use)
     4. Select unigrams (lemmas) and bi-grams (2 neighboring lemmas) from each sentence (Note: need to extract bigrams ONLY from a sentence, otherwise there will be erroneous bi-grams that happened to be one from one sentence and another from another sentence – this creates noise for the LDA model and reduces its accuracy)
     5. Lower case every extracted unigram and bigram – just in case
     6. Reduce the dimensionality of dictionary for topic modeling lemmas and bigrams (calculate lemma/bigram frequency in the texts for LDA, delete very infrequent and very frequent – you can use your choice of lower and upper-frequency boundaries, I like to use 1st and 99th percentiles)

1. Before training LDA model, you need to estimate the number of topics in a corpus (through topic coherence). - You can use “3\_step…” notebooks for reference
2. Create a document-term matrix for the LDA model using selected lemmas and bigrams in step (1)
3. Train the LDA model with corpus from (3) and a selected number of topics from (2)
4. You can examine how good topics are separated in the resulted LDA model through Interactive visualization of topics using pyLDAviz and repeat steps above if it is not to your liking
5. Repeat all steps for any modifications: using only lemmas, adding trigrams, etc.

How good a model is a subjective decision unless you have a set of good examples to use as a reference point in the model evaluation.

Notes

* LDA does not name topics – just gives the leading words in them (that you can see in the visualization.
* You can name resulting topics manually by reading the leading words in each topic or examining which documents have those topics as most probable (you can use function “topics” from the “3\_step…” notebook for that or modify step “Topic distribution across reviews” in the same notebook)

Python pandas syntaxis I use everywhere might be disturbing since it is new for you. In general “.apply(lambda…)” means that I apply the following calculations for each cell in a pandas data frame column – it is like looping through those values.

**Example:**

df\_data['doc2bow'].apply(lambda d: sorted(result\_lda\_model[d],key=lambda x:x[1],reverse=True))

for each document d :

1. takes a doc2bow vector
2. puts it in the LDA model result\_lda\_model
3. extracts list of topics with their probability (LDA model prediction/inference/output)
4. sort that list by topic probability in descending order

This example illustrates how you can get leading topics for each document in EUR-Lex and for user’ unseen phrases. Any unseen phrase needs to be preprocessed the same way as EUR-Lex documents so that the model can infer topics this phrase have.

After you get topics for each EUR-Lex document, you can

* cluster them as you do using DBSCAN or any other clustering algo.
* Get the representative topic vector (collection of topics) of that cluster
* Use the representative vector for cluster selection for an unseen phrase (that is, an unseen phrase is related to a cluster of documents if its topic vector is similar to the representative topic vector of that cluster, for example using cosign similarity measure)