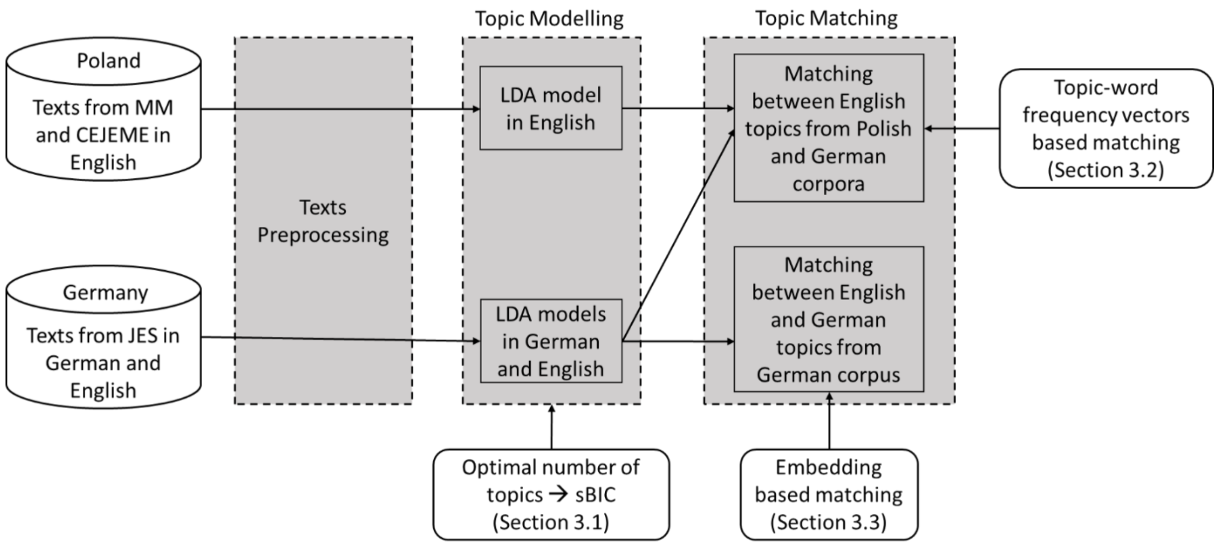
This document describes the code for the paper "Cross-corpora comparisons of topics and topic trends". The scripts are numbered according to the phases of the proposed project pipeline, which is presented below.


Topic Modelling
Topic Matching
Poland
Texts from MM and CEJEME in English
Germany
Texts from JES in German and English
Texts Preprocessing
LDA model in English
LDA models in German and English
Matching between English topics from Polish and German corpora
Matching between English and German topics from German corpus
Topic-word frequency vectors based matching (Section 3.2)
Optimal number of topics → sBIC (Section 3.1)
Embedding based matching (Section 3.3)

First, preprocessing of the text data was performed (see 01_ImportData_Preprocessing_GermanData.ipynb for the German texts collection and 01_Preprocessing&LDAPolishCorpora.ipynb for the Polish texts collection). Since the original texts could not be provided, you won't be able to execute these scripts. However, you can take a look at all the preprocessing steps that were performed (language detection, cleaning, lemmatizing) and packages that were used.

Second, to determine the optimal number of topics for each corpus, we implemented and applied sBIC routine (see the scripts beginning with "02_Topic_Number_Selection …"). You can run these scripts as document-term matrices for all the subsets are provided. To be able to run the proposed sBIC routine, please add the following functions to the file "...\sklearn\decomposition_lda.py" in the directory of scikit-learn module:

```
def loglik(self, X):
    """Calculate the expected sample log-likelihood.
    Parameters
    ----------
    X : array-like or sparse matrix, shape=(n_samples, n_features)
      Document word matrix.
    Returns
    -------
    score : float
    """
    X = self._check_non_neg_array(X, reset_n_features=True, whom="LatentDirichletAllocation.loglik")
    doc_topic_distr = self._unnormalized_transform(X)
    score = self._approx_loglik(X, doc_topic_distr, sub_sampling=False)
    return score
```

```
import math
def _approx_loglik(self, X, doc_topic_distr, sub_sampling):
    """Estimate the approximate sample log-likelihood
    Parameters
    ----------
    X : array-like or sparse matrix, shape=(n_samples, n_features)
        Document word matrix.
    doc_topic_distr : array, shape=(n_samples, n_components)
        Document topic distribution. In the literature, this is called
        gamma.
    sub_sampling : boolean, optional, (default=False)
        Compensate for subsampling of documents.
        It is used in calculate bound in online learning.
    Returns
    -------
    score : float
    """

    is_sparse_x = sp.issparse(X)
    n_samples, n_components = doc_topic_distr.shape
    n_features = self.components_.shape[1]
    score = 0
    dirichlet_doc_topic = _dirichlet_expectation_2d(doc_topic_distr)
    dirichlet_component_ = _dirichlet_expectation_2d(self.components_)
    doc_topic_prior = self.doc_topic_prior_
    topic_word_prior = self.topic_word_prior_
    if is_sparse_x:
        X_data = X.data
        X_indices = X.indices
        X_indptr = X.indptr
    # E[log p(docs | theta, beta)]
    for idx_d in range(0, n_samples):
        if is_sparse_x:
            ids = X_indices[X_indptr[idx_d]:X_indptr[idx_d + 1]]
            cnts = X_data[X_indptr[idx_d]:X_indptr[idx_d + 1]]
        else:
```

```
      ids = np.nonzero(X[idx_d, :])[0]
      cnts = X[idx_d, ids]
    temp = (dirichlet_doc_topic[idx_d, :, np.newaxis]
        + dirichlet_component_[:, ids])
    norm_phi = logsumexp(temp, axis=0)
    score += np.dot(cnts, norm_phi)
    scale=0
    for i in range(1,np.sum(cnts)+1):
      scale +=math.log(i)
    for i in range(0,len(cnts)):
      for j in range(1,cnts[i]+1):
        scale -=math.log(j)
    score +=scale
  return score
```

Third, topic matching was performed using the scripts beginning with "03":

- 03.1_Emmbeddings_based_Matching_Germany.ipynb: this script can be executed since all the needed source files are provided. However, you would need to download the pre-trained word embeddings and change the path name in the script. Here, the matching between DE^ENG and DE^GER using pre-trained multilingual word embeddings is performed.
- 03.2_Topic_word_frequency_based_Matching.ipynb: Here, the matching between PL^ENG and DE^ENG is performed based on topic-word frequencies.
- 03.3_Matched_Topics_Poland_Germany.ipynb: The matches between all the three data subsets are constructed and plotted, as presented in Figures 11 and 12. You can run the script when you run the scripts 03.1 and 03.2 and adjust the paths where you want to store the results.

Apart from the main results, results on robustness checks are presented in the paper:

- Robustness_Machine_Translation.ipynb: the script is executable beginning from 3.2, subsection sBIC routine. All the further steps, matching with the Polish corpus can be also executed. The results are reported in Appendix C.1.
- Robustness_sklearn_vs_gensim.ipynb: this script compares the LDA models trained using gensim and sklearn modules. The results of this robustness check are presented in Appendix C.2.
- Robustness_Jensen_Shannon.ipynb: this script performs topic-word frequencies-based matching using a different similarity measure. The results of this robustness check are presented in Appendix C.3.
- Other_Evaluation_Metrics.ipynb: this script can be used to calculate other evaluation metrics widely used in the literature. The results are presented in Figure 5.