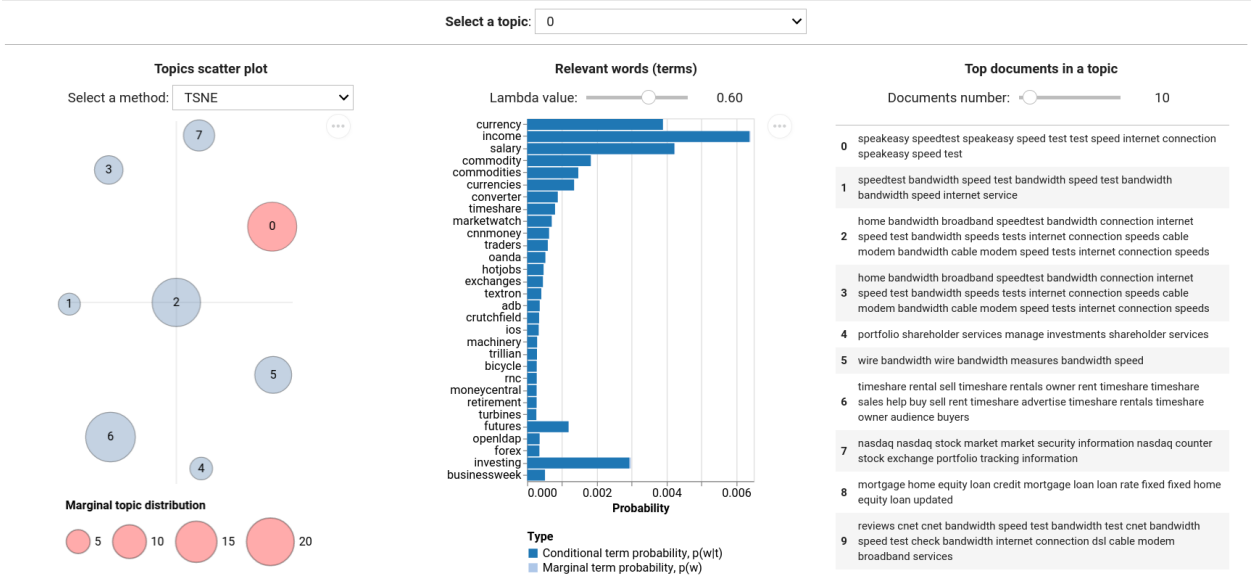
tmplot

Maksim Terpilowski

Jan 26, 2024

USAGE

tmplot is a Python package for visualizing topic modeling results. It provides the interactive report interface that borrows much from LDAvis/pyLDAvis and builds upon it offering a number of metrics for calculating topics distances and a number of algorithms for calculating scatter coordinates of topics.



CHAPTER

ONE

FEATURES

- Supported models:
  - tomotopy: `LDAModel`, `LLDAModel`, `CTModel`, `DMRModel`, `HDPModel`, `PTModel`, `SLDAModel`, `GDMRModel`
  - gensim: `LdaModel`, `LdaMulticore`
  - bitermplus: `BTM`
- Supported distance metrics:
  - Kullback-Leibler (symmetric and non-symmetric) divergence
  - Jenson-Shannon divergence
  - Jeffrey's divergence
  - Hellinger distance
  - Bhattacharyya distance
  - Total variation distance
  - Jaccard inversed index
- Supported algorithms for calculating topics scatter coordinates:
  - t-SNE
  - SpectralEmbedding
  - MDS
  - LocallyLinearEmbedding
  - Isomap

1.1 Installation

The package can be installed from PyPi:

```
pip install tmplot
```

Or directly from this repository:

```
pip install git+https://github.com/maximtrp/tmplot.git
```

1.1.1 Dependencies

- `numpy`
- `scipy`
- `scikit-learn`
- `pandas`
- `altair`
- `ipywidgets`
- `tomotopy`, `gensim`, and `bitermplus` (for topic modeling)

1.2 Tutorial

1.2.1 Importing packages

```
import warnings
warnings.filterwarnings('ignore')
```

```
import tmplot as tmp
import pickle as pkl
import pandas as pd
```

1.2.2 Importing data

Let's take the BTM model trained on a test dataset (*SearchSnippets*) as an example. We will begin with reading it from a file:

```
with open('data/model_btm.pkl', 'rb') as file:
    model = pkl.load(file)
```

```
docs = pd.read_csv('data/SearchSnippets.txt.gz', header=None).values.ravel()
```

1.2.3 Matrices

Researchers working with topic models often need to obtain `phi` (words vs topics probability) and `theta` (topics vs documents probability) matrices. *Tmplot* provides two functions for getting these matrices from `tomotopy`, `bitermplus`, and `gensim` models.

Phi matrix

Note that you will need to pass a vocabulary for a gensim model.

```
phi = tmp.get_phi(model)
phi.head()
```

```
topics                    0             1             2             3  \
words
aaa            3.195102e-08  3.012856e-08  3.047842e-08  3.542745e-08
aaas           3.837318e-05  3.012856e-08  3.047842e-08  3.542745e-08
aaron          3.195102e-08  3.012856e-08  3.047842e-08  3.542745e-08
aau            3.195102e-08  3.012856e-08  3.047842e-08  3.542745e-08
abbreviations  7.990951e-05  3.163800e-04  3.047842e-08  3.542745e-08

topics                    4             5             6             7
words
aaa            3.836165e-08  2.961217e-08  2.362519e-08  4.831267e-08
aaas           3.836165e-08  5.922729e-04  6.144912e-05  2.903592e-05
aaron          4.296888e-04  2.961217e-08  2.362519e-08  4.831267e-08
aau            3.836165e-08  2.961217e-08  2.362519e-08  4.203686e-04
abbreviations  3.836165e-08  2.961217e-08  2.386144e-06  4.831267e-08
```

Theta matrix

```
tmp.get_theta(model).head()
```

```
docs         0         1         2         3         4         5         6    \
topics
0       0.354702  0.294777  0.178074  0.332888  0.596412  0.726975  0.099094
1       0.000245  0.007173  0.021324  0.019411  0.029472  0.008740  0.011804
2       0.003073  0.057144  0.013837  0.014514  0.011813  0.002588  0.000247
3       0.003678  0.029281  0.010010  0.001287  0.027349  0.004351  0.018189
4       0.000927  0.035162  0.001736  0.319421  0.024606  0.042996  0.019524

docs         7         8         9    ...       990       991       992  \
topics                                ...
0       0.257602  0.532725  0.471059  ...  0.007651  0.085897  0.025840
1       0.036323  0.011349  0.003909  ...  0.069988  0.263869  0.058431
2       0.027391  0.002325  0.005435  ...  0.007558  0.014669  0.014206
3       0.085879  0.011453  0.002965  ...  0.007010  0.022462  0.007516
4       0.036119  0.001910  0.039332  ...  0.016587  0.056386  0.005925

docs         993       994       995       996       997       998       999
topics
0       0.019194  0.033898  0.020408  0.030728  0.036133  0.084323  0.024301
1       0.227196  0.022920  0.021660  0.040932  0.060534  0.150018  0.071271
2       0.002697  0.008854  0.017299  0.014710  0.027672  0.061375  0.011318
3       0.006018  0.001193  0.007400  0.007335  0.021119  0.012309  0.006168
4       0.003503  0.001620  0.006468  0.004151  0.018374  0.008712  0.087364

[5 rows x 1000 columns]
```

1.2.4 Documents

Here is how you can get documents with maximum probabilities $P(t|d)$ for each topic:

```
tmp.get_top_docs(docs, model=model)
```

```
                                              topic0  \
0  speakeasy speedtest speakeasy speed test test ...
1  speedtest bandwidth speed test bandwidth speed...
2  home bandwidth broadband speedtest bandwidth c...
3  home bandwidth broadband speedtest bandwidth c...
4  portfolio shareholder services manage investme...

                                              topic1  \
0  links jstor sici sici jstor postwar consumptio...
1  econpapers repec article econpapers postwar co...
2  findarticles articles consumption consumer exp...
3  financial financial international health insur...
4  consumption consumer rights consumption consum...

                                              topic2  \
0  imdb name julia roberts julia roberts imdb mov...
1  celebrities cruise celebrity tom cruise tom cr...
2  imdb name tom cruise tom cruise imdb movies ce...
3  absolutely roberts absolutely julia roberts ph...
4                     imdb title imdb movies celebs

                                              topic3  \
0               guitars bodies amps guitars strings
1  louis french fashion designer designer manufac...
2  fashion designers default fashion designers fa...
3  fashion designers audio fashion designer net f...
4  fashion fashion designers fashion designers fa...

                                              topic4  \
0  vcic unc edu vcic venture capital investment c...
1  national venture capital association foster un...
2  san jose mercury news venture capital expanded...
3  seattlepi nwsource venture seattle venture cap...
4  venture capital journal listening model ventur...

                                              topic5  \
0  washington edu drivers device drivers device d...
1               manufactures parallel serial drives
2  leonardo leonardo vinci inventor information c...
3  journals searching biomedical journals engine ...
4  lwn articles driver lwn device drivers kernel ...

                                              topic6  \
0  apache api dom document document xml standard ...
1  schools dom default xml dom tutorial xml docum...
2                          access cards ieee access
3  generator xml generator sample xml instance do...
4  reference standard template library standard t...
```

(continues on next page)

(continued from previous page)

```
                                                topic7
0  hypotheses hypotheses author illustrates hypot...
1                                surreal surreal
2  allposters surrealism posters surrealism poste...
3  hypotheses hypotheses nature research hypothes...
4  allposters beatles posters beatles prints allp...
```

1.2.5 Visualization

tmplot takes much from LDAvis, but also extends the functionality with a number of algorithms and metrics for plotting topics and terms. *tmplot* is based on ipywidgets and Altair (Vega-backed package for nice plots).

Topics

First, we need to calculate the coordinates of topics based on intertopic distance values. By default, the combination of *t-distributed Stochastic Neighbor Embedding* and *symmetric Kullback-Leibler divergence* is used to calculate topics coordinates in 2D, but a number of other metrics and algorithms are also available (see `tmplot.get_topics_dist` and `tmplot.get_topics_scatter` functions for additional information).

```
topics_coords = tmp.prepare_coords(model)
topics_coords.head()
```

```
           x          y  topic       size  label
0 -41.183987 -30.480648      0  21.160233      0
1 -11.704910 -34.631725      1   4.265470      1
2 -56.292171  -4.832846      2  20.599346      2
3   9.921317 -14.181945      3   7.176289      3
4 -45.702721  22.987968      4   4.535249      4
```

Plotting topics:

```
tmp.plot_scatter_topics(topics_coords, size_col='size', label_col='label')
```

```
alt.LayerChart(...)
```

Words (or terms)

tmplot also uses terms *relevance* that was introduced by Sievert and Shirley (2014) for sorting terms.

```
terms_probs = tmp.calc_terms_probs_ratio(phi, topic=0, lambda_=1)
```

```
tmp.plot_terms(terms_probs)
```

```
alt.Chart(...)
```

Documents

```
top_docs_topic0 = tmp.get_top_docs(docs, model=model, docs_num=2, topics=[0])
top_docs_topic0
```

```
                                                topic0
0  speakeasy speedtest speakeasy speed test test ...
1  speedtest bandwidth speed test bandwidth speed...
```

The following output is used within the interactive interface that we will explore shortly:

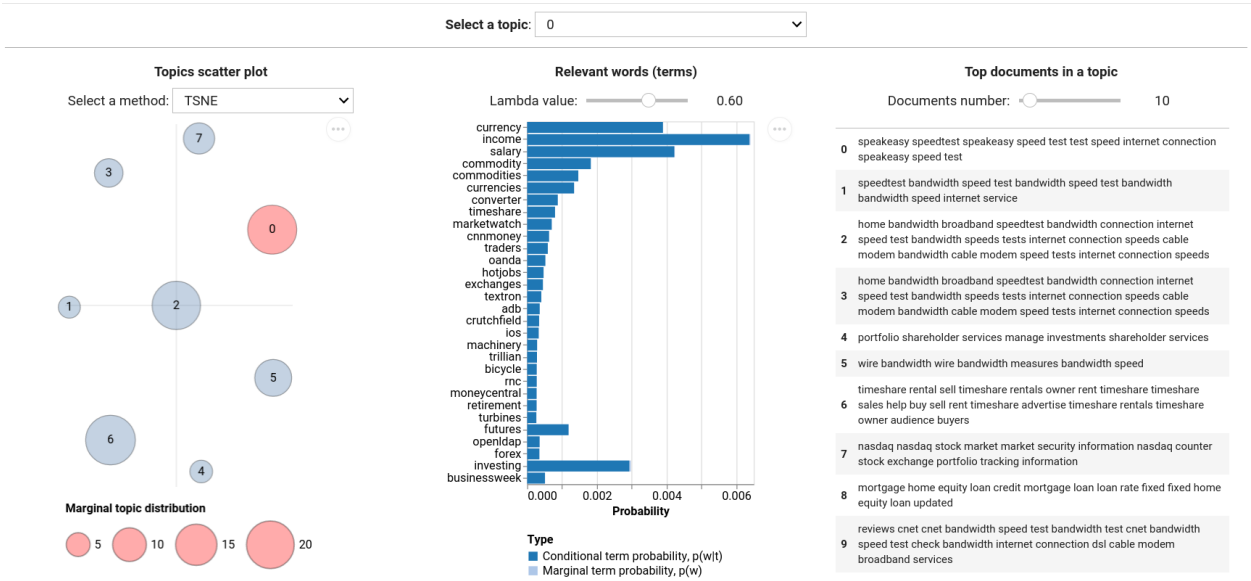
```
tmp.plot_docs(top_docs_topic0)
```

```
<IPython.core.display.HTML object>
```

1.2.6 Interactive report interface

To run the report interface, just call `tmplot.report()` function with your model and docs. You can tweak most of the hidden parameters using keyword arguments (see function docstring).

```
tmp.report(model, docs=docs, height=400, width=250)
```



1.3 Helper functions

`tmplot.`**`get_phi`**(*model: object*, *vocabulary: Sequence | None = None*) → DataFrame

Get words vs topics matrix (phi).

Returns `phi` matrix of shape W x T, where W is the number of words, and T is the number of topics.

Parameters

- **`model`** (*`object`*) – Topic model instance.
- **`vocabulary`** (*`Optional[Sequence], optional`*) – Vocabulary as a list of words. Needed for getting `phi` matrix from `gensim` model instance.

Returns
Words vs topics matrix (phi).

Return type
pandas.DataFrame

tmplot.**get_theta**(*model: object*, *corpus: List | None = None*) → DataFrame

Get topics vs documents (theta) matrix.

Returns theta matrix of shape T x D, where T is the number of topics, D is the number of documents.

Parameters

- **model** (*object*) – Topic model instance.
- **corpus** (*Optional[List], optional*) – Corpus.

Returns
Topics vs documents matrix (theta).

Return type
pandas.DataFrame

tmplot.**get_docs**(*model: object*) → List[str]

Retrieve documents from topic model object.

Parameters
model (*object*) – Topic model instance.

Returns
List of documents.

Return type
List[str]

tmplot.**get_top_docs**(*docs: Sequence[str]*, *model: object = None*, *theta: ndarray = None*, *corpus: List | None = None*, *docs_num: int = 5*, *topics: Sequence[int] = None*) → DataFrame

Get top documents for all (or a selected) topic.

Parameters

- **docs** (*Sequence*) – List of documents.
- **model** (*object, optional*) – Topic model instance.
- **theta** (*numpy.ndarray, optional*) – Topics vs documents matrix.
- **corpus** (*Optional[List], optional*) – Corpus for gensim model.
- **docs_num** (*int, optional*) – Number of documents to return.
- **topics** (*Sequence[int], optional*) – Sequence of topics indices.

Returns
Top documents.

Return type
pandas.DataFrame

Raises
ValueError – If neither a model or theta matrix is passed, ValueError is raised.

tmplot.**get_relevant_terms**(*phi: ndarray | DataFrame*, *topic: int*, *lambda_: float = 0.6*) → Series

Select relevant terms.

Parameters

- **phi** (*Union[numpy.ndarray, pandas.DataFrame]*) – Words vs topics matrix (phi).
- **topic** (*int*) – Topic index.
- **lambda** (*float = 0.6*) – Weight parameter. It determines the weight given to the probability of term W under topic T relative to its lift[2]. Setting it to 1 equals topic-specific probabilities of terms.

References

Returns
Terms sorted by relevance (descendingly).

Return type
pandas.Series

tmplot.**get_salient_terms**(*terms_freqs: ndarray*, *phi: ndarray*, *theta: ndarray*) → ndarray

Get salient terms.

Calculated as: saliency(w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))], where `w` is a term index, `t` is a topic index.

Parameters

- **terms_freqs** (*numpy.ndarray*) – Words frequencies.
- **phi** (*numpy.ndarray*) – Words vs topics matrix.
- **theta** (*numpy.ndarray*) – Topics vs documents matrix.

Returns
Terms saliency values.

Return type
numpy.ndarray

tmplot.**calc_terms_marg_probs**(*phi: ndarray | DataFrame*, *word_id: int | None = None*) → ndarray | Series

Calculate marginal terms probabilities.

Parameters

- **phi** (*Union[numpy.ndarray, pandas.DataFrame]*) – Words vs topics matrix.
- **word_id** (*Optional[int]*) – Word index.

Returns
Marginal terms probabilities.

Return type
Union[numpy.ndarray, pandas.Series]

tmplot.**calc_topics_marg_probs**(*theta: DataFrame | ndarray*, *topic_id: int = None*) → DataFrame | ndarray

Calculate marginal topics probabilities.

Parameters

[2] Sievert, C., & Shirley, K. (2014). LDAvis: A method for visualizing and interpreting topics. In Proceedings of the workshop on interactive language learning, visualization, and interfaces (pp. 63-70).

- **theta** (*Union[pandas.DataFrame, numpy.ndarray]*) – Topics vs documents matrix.
- **topic_id** (*int, optional*) – Topic index.

Returns
Marginal topics probabilities.

Return type
Union[pandas.DataFrame, numpy.ndarray]

tmplot.**calc_terms_probs_ratio**(*phi: DataFrame*, *topic: int*, *terms_num: int = 30*, *lambda_: float = 0.6*) → DataFrame

Get terms conditional and marginal probabilities.

Parameters

- **phi** (*pandas.DataFrame*) – Words vs topics matrix.
- **topic** (*int*) – Topic index.
- **terms_num** (*int, optional*) – Number of words to return.
- **lambda** (*float, optional*) – Weight parameter. It determines the weight given to the probability of term W under topic T relative to its lift[1]. Setting it to 1 equals topic-specific probabilities of terms.

References

Returns
Words conditional and marginal probabilities.

Return type
pandas.DataFrame

1.4 Distance functions

tmplot.**get_topics_dist**(*phi: ndarray | DataFrame*, *method: str = 'sklb'*, ***kwargs*) → ndarray

Finding closest topics in models.

Parameters

- **phi** (*Union[ndarray, DataFrame]*) – Words vs topics matrix (W x T).
- **method** (*str = "sklb"*) – Comparison method. Possible variants: 1) “klb” - Kullback-Leibler divergence. 2) “sklb” - Symmetric Kullback-Leibler divergence. 3) “jsd” - Jensen-Shannon divergence. 4) “jef” - Jeffrey’s divergence. 5) “hel” - Hellinger distance. 6) “bhat” - Bhattacharyya distance. 7) “tv” — Total variation distance. 8) “jac” - Jaccard index.
- ****kwargs** (*dict*) – Keyword arguments passed to distance function.

Returns
Topics distances matrix.

Return type
numpy.ndarray

[1] Sievert, C., & Shirley, K. (2014). LDAvis: A method for visualizing and interpreting topics. In Proceedings of the workshop on interactive language learning, visualization, and interfaces (pp. 63-70).

tmplot.**get_topics_scatter**(*topic_dists: ndarray*, *theta: ndarray*, *method: str = 'tsne'*, *method_kws: dict = None*) → DataFrame

Calculate topics coordinates for a scatter plot.

Parameters

- **topic_dists** (*numpy.ndarray*) – Topics distance matrix.
- **theta** (*numpy.ndarray*) – Topics vs documents probability matrix.
- **method** (*str = 'tsne'*) – Method to calculate topics scatter coordinates (X and Y). Possible values: 1) 'tsne' - t-distributed Stochastic Neighbor Embedding. 2) 'sem' - SpectralEmbedding. 3) 'mds' - MDS. 4) 'lle' - LocallyLinearEmbedding. 5) 'ltsa' - LocallyLinearEmbedding with LTDA method. 6) 'isomap' - Isomap.
- **method_kws** (*dict = None*) – Keyword arguments passed to method function.

Returns
Topics scatter coordinates.

Return type
DataFrame

tmplot.**get_top_topic_words**(*phi: DataFrame*, *words_num: int = 20*, *topics_idx: List[int] | ndarray = None*) → DataFrame

Select top topic words from a fitted model.

Parameters

- **phi** (*pandas.DataFrame*) – Words vs topics matrix (phi) with words as indices and topics as columns.
- **words_num** (*int = 20*) – The number of words to select.
- **topics_idx** (*Union[List, numpy.ndarray] = None*) – Topics indices.

Returns
Words with highest probabilities in all (or selected) topics.

Return type
DataFrame

1.5 Visualization

tmplot.**report**(*model: object*, *docs: Sequence[str]*, ***, *topics_labels: Sequence[str] | None = None*, *corpus: List | None = None*, *layout: Layout = None*, *show_headers: bool = True*, *show_docs: bool = True*, *show_words: bool = True*, *show_topics: bool = True*, *topics_kws: dict = None*, *height: int = 500*, *width: int = 300*, *coords_kws: dict = None*, *words_kws: dict = None*, *docs_kws: dict = None*, *top_docs_kws: dict = None*) → VBox

Interactive report interface.

Parameters

- **model** (*object*) – Topic model instance.
- **docs** (*Sequence[str]*) – Documents.
- **topics_labels** (*Optional[Sequence[str]], optional*) – Topics labels.
- **corpus** (*Optional[List[str]], optional*) – Gensim corpus.

- **layout** (*wdg.Layout, optional*) – Interface layout instance.
- **show_headers** (*bool, optional*) – Show headers.
- **show_docs** (*bool, optional*) – Show documents widget.
- **show_words** (*bool, optional*) – Show words widget.
- **show_topics** (*bool, optional*) – Show topics scatter plot widget.
- **topics_kws** (*dict, optional*) – Keyword arguments passed to *tmplot.plot_scatter_topics()*.
- **coords_kws** (*dict, optional*) – Keyword arguments passed to *tmplot.prepare_coords()*.
- **words_kws** (*dict, optional*) – Keyword arguments passed to *tmplot.plot_terms()*.
- **docs_kws** (*dict, optional*) – Keyword arguments passed to *tmplot.plot_docs()*.
- **top_docs_kws** (*dict, optional*) – Keyword arguments passed to *tmplot.get_top_docs()*.

Returns
Report interface as a VBox instance.

Return type
ipywidgets.widgets.widget_box.VBox

tmplot.**prepare_coords**(*model: object*, *labels: Sequence | None = None*, *dist_kws: dict = None*, *scatter_kws: dict = None*) → DataFrame

Prepare coordinates for topics scatter plot.

Parameters

- **model** (*object*) – Topic model instance.
- **labels** (*Optional[Sequence]*) – Topics labels.
- **dist_kws** (*dict, optional*) – Keyword arguments passed to *tmplot.get_topics_dist()*.
- **scatter_kws** (*dict, optional*) – Keyword arguments passed to *tmplot.get_topics_scatter()*.

1.6 Metrics

tmplot.**entropy**(*phi: ndarray*, *max_probs: bool = False*)

Renyi entropy calculation routine[1].

Renyi entropy can be used to estimate the optimal number of topics: fit several models varying the number of topics and choose the model for which Renyi entropy is minimal.

Parameters
phi (*np.ndarray*) – Topics vs words probabilities matrix (T x W).

Returns

- **renyi** (*double*) – Renyi entropy value.

[1] Koltcov, S. (2018). Application of Rényi and Tsallis entropies to topic modeling optimization. Physica A: Statistical Mechanics and its Applications, 512, 1192-1204.

- **max_probs** (*bool*) – Use maximum probabilities of terms per topics instead of all probability values.

References

Example

```
>>> import tmplot as tmp
>>> # Preprocessing step
>>> # ...
>>> # Model fitting step
>>> # model = ...
>>> # phi = ...
>>> # Entropy calculation
>>> entropy = tmp.entropy(phi)
```

1.7 Stability functions

tmplot.**get_closest_topics**(*models: List[Any]*, *ref: int = 0*, *method: str = 'sklb'*, *top_words: int = 100*, *verbose: bool = True*) → Tuple[ndarray, ndarray]

Finding closest topics in models.

Parameters

- **models** (*List[Any]*) – List of supported and fitted topic models.
- **ref** (*int = 0*) – Index of reference matrix (zero-based indexing).
- **method** (*str = "sklb"*) – Distance calculation method. Possible variants: 1) “klb” - Kullback-Leibler divergence. 2) “sklb” - Symmetric Kullback-Leibler divergence. 3) “jsd” - Jensen-Shannon divergence. 4) “jef” - Jeffrey’s divergence. 5) “hel” - Hellinger distance. 6) “bhat” - Bhattacharyya distance. 7) “tv” - Total variation distance. 8) “jac” - Jaccard index.
- **top_words** (*int = 100*) – Number of top words in each topic to use in Jaccard index calculation.
- **verbose** (*bool = True*) – Verbose output (progress bar).

Returns

- **closest_topics** (*np.ndarray*) – Closest topics indices in one two-dimensional array (topics × models). Columns correspond to the compared models (their indices), rows are the closest topics pairs.
- **dist** (*np.ndarray*) – Closest topics distances (e.g., Kullback-Leibler or Jaccard index values). Shape of this array corresponds to the shape of the first returned argument.

Example

```
>>> # `models` must be an iterable of fitted models
>>> closest_topics, kldiv = tmplot.get_closest_topics(models)
```

tmplot.**get_stable_topics**(*closest_topics: ndarray*, *dist: ndarray*, *norm: bool = True*, *inverse: bool = True*, *inverse_factor: float = 1.0*, *ref: int = 0*, *thres: float = 0.9*, *thres_models: int = 2*) → Tuple[ndarray, ndarray]

Finding stable topics in models.

Parameters

- **closest_topics** (*np.ndarray*) – Closest topics indices in a two-dimensional array. Columns correspond to the compared matrices (their indices), rows are the closest topics pairs. Typically, this should be the first value returned by *tmplot.get_closest_topics()* function.
- **dist** (*np.ndarray*) – Distance values: Kullback-Leibler divergence or Jaccard index values corresponding to the matrix of the closest topics. Typically, this should be the second value returned by *tmplot.get_closest_topics()* function.
- **norm** (*bool = True*) – Normalize distance values (passed as `dist` argument).
- **inverse** (*bool = True*) – Inverse distance values by subtracting them from `inverse_factor`. Should be set to `False` if Jaccard index was used to calculate closest topics.
- **inverse_factor** (*float = 1.0*) – Subtract distance values from this factor to inverse.
- **ref** (*int = 0*) – Index of reference matrix (i.e. reference column index, zero-based indexing).
- **thres** (*float = 0.9*) – Threshold for distance values filtering.
- **thres_models** (*int = 2*) – Minimum topic recurrence frequency across all models.

Returns

- **stable_topics** (*np.ndarray*) – Filtered matrix of the closest topics indices (i.e. stable topics).
- **dist** (*np.ndarray*) – Filtered distance values corresponding to the matrix of the closest topics.

See also:

tmplot.get_closest_topics

Example

```
>>> closest_topics, kldiv = tmplot.get_closest_topics(models)
>>> stable_topics, stable_kldiv = tmplot.get_stable_topics(
...     closest_topics, kldiv)
```

1.8 Visualization

tmplot.**plot_scatter_topics**(*topics_coords: ndarray | DataFrame*, *x_col: str = 'x'*, *y_col: str = 'y'*, *topic: int = None*, *size_col: str = None*, *label_col: str = None*, *color_col: str = None*, *topic_col: str = None*, *font_size: int = 13*, *x_kws: dict = None*, *y_kws: dict = None*, *chart_kws: dict = None*, *circle_kws: dict = None*, *circle_enc_kws: dict = None*, *text_kws: dict = None*, *text_enc_kws: dict = None*, *size_kws: dict = None*, *color_kws: dict = None*) → Chart

Topics scatter plot in 2D.

Parameters

- **topics_coords** (*Union[ndarray, DataFrame]*) – Topics scatter coordinates.
- **x_col** (*str, optional*) – X column name.
- **y_col** (*str, optional*) – Y column name.
- **topic** (*int, optional*) – Topic index.
- **size_col** (*str, optional*) – Column with size values.
- **label_col** (*str, optional*) – Column with topic labels.
- **color_col** (*str, optional*) – Column with colors.
- **topic_col** (*str, optional*) – Column with topics texts.
- **font_size** (*int, optional*) – Font size.
- **x_kws** (*dict, optional*) – Keyword arguments passed to `altair.X()`.
- **y_kws** (*dict, optional*) – Keyword arguments passed to `altair.Y()`.
- **chart_kws** (*dict, optional*) – Keyword arguments passed to `altair.Chart()`.
- **circle_kws** (*dict, optional*) – Keyword arguments passed to `altair.Chart.mark_circle()`.
- **circle_enc_kws** (*dict, optional*) – Keyword arguments passed to `altair.Chart.encode()` for circle elements.
- **text_kws** (*dict, optional*) – Keyword arguments passed to `altair.Chart.mark_text()`.
- **text_enc_kws** (*dict, optional*) – Keyword arguments passed to `altair.Chart.encode()` for text elements.
- **size_kws** (*dict, optional*) – Keyword arguments passed to `altair.Size()` for circle elements.
- **color_kws** (*dict, optional*) – Keyword arguments passed to `altair.Color()` for circle elements.

Returns
Topics scatter plot.

Return type
altair.Chart

tmplot.**plot_terms**(*terms_probs: DataFrame*, *x_col: str = 'Probability'*, *y_col: str = 'Terms'*, *color_col: str = 'Type'*, *font_size: int = 13*, *chart_kws: dict = None*, *bar_kws: dict = None*, *x_kws: dict = None*, *y_kws: dict = None*, *color_kws: dict = None*) → Chart

Plot words conditional and marginal probabilities.

Parameters

- **terms_probs** (*DataFrame*) – Words probabilities.
- **x_col** (*str, optional*) – X column name.
- **y_col** (*str, optional*) – Y column name.
- **color_col** (*str, optional*) – Column with values types (for coloring).
- **font_size** (*int, optional*) – Font size.
- **chart_kws** (*dict, optional*) – Keyword arguments passed to `altair.Chart()`.
- **bar_kws** (*dict, optional*) – Keyword arguments passed to `altair.Chart.mark_bar()`.
- **x_kws** (*dict, optional*) – Keyword arguments passed to `altair.X()`.
- **y_kws** (*dict, optional*) – Keyword arguments passed to `altair.Y()`.
- **color_kws** (*dict, optional*) – Keyword arguments passed to `altair.Color()`.

Returns
Terms probabilities chart.

Return type
altair.Chart

tmplot.**plot_docs**(*docs: Sequence[str] | DataFrame*, *styles: str = None*, *html_kws: dict = None*) → DataFrame

Documents plotting functionality for report interface.

Parameters

- **docs** (*Union[Sequence[str], DataFrame]*) – Documents.
- **styles** (*str, optional*) – Styles string for formatting the table with documents. Concatenated with HTML.
- **html_kws** (*dict, optional*) – Keyword arguments passed to `pandas.DataFrame.to_html()` method.

Returns
Topic documents.

Return type
ipywidgets.HTML

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX