Eksploracja Krajowych planów na rzecz energii i klimatu

Co wyniknęło z eksploracji?

- Głównym krokiem przed eksploracją było przetworzenie danych w odpowiedni sposób. Uwzględniona została przy tym zakładana struktura dokumentów. Okazało się, że rzeczywiście występują istotne różncie pomiędzy poszczególnymi sekcjami i wymiarami, zatem analiza w podziale na składowe dokumentu ma sens.
- Okazało się również, że w danych widać różnice pomiędzy dokumentami poszczególnych państw, co także stanowi dobry znak dla dalszej pracy zakładającej dokładniejsze porównania między poszczególnymi członkami UE.
- Pojawiły się kolejne pytania badawcze, np. dotyczące różnic w traktowaniu o transporcie w procesie dekarbonizacji.
- Zidentyfikowano słowa, które należy rozważyć w kontekście uwzględnienia jako stop-słowa.
- Wskazano dalsze kroki: próba poprawy odczytu tekstów z PDF (bez tabel, wykresów, numeracji stron)

Importy

```
In [1]: # from google.colab import drive
        # drive.mount('/content/drive')
```

```
In [2]: # ! pip install swifter
        # ! pip install matplotlib==3.4.0
        # ! pip install textacy
        # ! pip install thinc
        # ! pip install gensim
        # ! pip install pyLDAvis
```

```
In [3]: # !python -m spacy download en_core_web_lg
        # # trzeba uruchomić ponownie środowisko wykonawcze po pobraniu
```

```
In [4]: import pandas as pd
        import numpy as np
        import spacy
        from gensim.corpora.dictionary import Dictionary
        from gensim.models.ldamulticore import LdaMulticore

        import pyLDAvis.gensim_models
        pyLDAvis.enable_notebook()
```

```
/usr/local/lib/python3.7/dist-packages/past/types/oldstr.py:5: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from 'collections.abc' is dep
ated since Python 3.3,and in 3.9 it will stop working
  from collections import Iterable
```

```
In [5]: en = spacy.load("en_core_web_lg")
```

```
In [6]: import os
        import pickle
        from collections import Counter
        from tqdm import tqdm
        import seaborn as sns
        import matplotlib.pyplot as plt
        sns.set_theme(style="whitegrid")
        import plotly.express as px
```

```
/usr/local/lib/python3.7/dist-packages/yaml/constructor.py:126: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from 'collections.abc' is de
cated since Python 3.3,and in 3.9 it will stop working
  if not isinstance(key, collections.Hashable):
/usr/local/lib/python3.7/dist-packages/dask/array/numpy_compat.py:21: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warnin
use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  np.divide(0.4, 1, casting="unsafe", dtype=np.float),
/usr/local/lib/python3.7/dist-packages/scipy/io/matlab/mio5.py:98: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this warning, us
bool` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  from .mio5_utils import VarReader5
```

```
In [7]: import plotly.offline as py
        py.init_notebook_mode()
```

```
In [8]: #DIR = '/content/drive/MyDrive/NLP-klimat/'
```

```
In [9]: DIR = '../NLP-klimat/'
```

```
In [ ]:
```

Wczytanie danych

Teksty zostały odczytane z PDF-ów na podstawie wcześniejszego otagowania poszczególnych dokumentów.

```
In [10]: NECP_annotations = pd.read_csv(DIR+'NECP.txt')
```

```
In [11]: NECP_annotations = NECP_annotations.replace({"None": None})
```

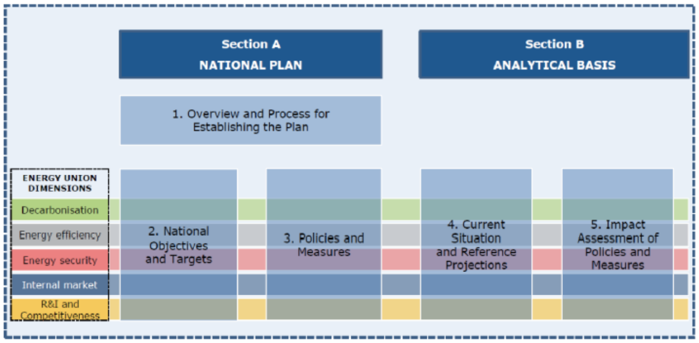
Intro: czym jest NECP?

NECP - National Energy and Climate Plan (Krajowy plan na rzecz energii i klimatu)

Aby zrealizować ustanowione przez Unię Europejską cele w zakresie energii i klimatu na 2030 rok, państwa członkowskie zostały zobowiązane do ustanowienia 10-letniego planu na rzecz energii i klimatu na okres od 2021 do 2030 roku (NECP).

Struktura NECP

Figure 4. Structure of NECPs according to Regulation (EU) 2018/1999 Annex I



Zatem dla każdego z 27 państw członkowskich otrzymujemy sekcje:

- Overview and Process for Establishing the Plan - Zarys ogólny i proces opracowywania planu
- National Objectives and Targets - Krajowe założenia i cele
- Policies and Measures - Polityki i działania
- Current Situation and Reference Projections - Aktualna sytuacja i prognozy z obecną polityką klimatyczną
- Impact Assessment of Planned Policies and Measures - Ocena wpływu planowanych działań na rzecz klimatu

Według wzorcowej struktury sekcje 2-5 powinny byc podzielone na 5 wymiarów:

- Decarbonisation - Obniżenie emisyjności
- Energy efficiency - Efektywność energetyczna
- Energy security - Bezpieczeństwo energetyczne
- Internal market - Wewnętrzny rynek energii
- R&I and Competitiveness - Badania naukowe, innowacje i konkurencyjność

W rzeczywistości w większości planów w sekcji oceny wpływu planowanych działań na rzecz klimatu nie ma podziału na 5 wymiarów

```
In [12]: necp_processed = pd.read_csv(DIR+'necp_processed.csv', index_col = 0)
```

Kolumny zaimportowanej ramki danych.

```
In [13]: necp_processed.columns

Out[13]: Index(['country', 'file_name', 'subsection', 'energy_union_dimension',
                'start_page', 'end_page', 'start_text', 'end_text', 'text'],
               dtype='object')
```

```
In [14]: necp_processed.drop(['start_page', 'end_page', 'start_text', 'end_text'], axis = 1, inplace = True)
```

```
In [15]: necp_processed.drop(necp_processed[necp_processed.isnull()["text"]].index, axis = 0, inplace = True)
```

```
In [16]: len(necp_processed)

Out[16]: 453
```

Zostało 453 części dokumentów.

Przetworzenie tekstów

```
In [17]: import swifter
         import warnings
         warnings.filterwarnings("default")

         /usr/local/lib/python3.7/dist-packages/numba/core/types/__init__.py:108: DeprecationWarning:

         `np.long` is a deprecated alias for `np.compat.long`. To silence this warning, use `np.compat.long` by itself. In the likely event your code does not need to work on Pyth
         2 you can use the builtin `int` for which `np.compat.long` is itself an alias. Doing this will not modify any behaviour and is safe. When replacing `np.long`, you may wis
         o use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.
         Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

         /usr/local/lib/python3.7/dist-packages/numba/core/types/__init__.py:109: DeprecationWarning:

         `np.long` is a deprecated alias for `np.compat.long`. To silence this warning, use `np.compat.long` by itself. In the likely event your code does not need to work on Pyth
         2 you can use the builtin `int` for which `np.compat.long` is itself an alias. Doing this will not modify any behaviour and is safe. When replacing `np.long`, you may wis
         o use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.
         Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
```

```
In [18]: # tqdm.pandas()
         # necp_docs = necp_processed['text'].swifter.apply(en)
```

```
In [19]: # # eksport przetworzonych dokumentów
         # with open(DIR + 'necp_docs_lg.pickle', 'wb') as f:
         #   pickle.dump(necp_docs, f)
```

Wczytanie

```
In [20]: # import przetworzonych dokumentów
         with open(DIR + 'necp_docs_lg.pickle', 'rb') as f:
             necp_docs_2 = pickle.load(f)
```

```
In [21]: necp_docs = necp_docs_2
```

```
In [22]: countries_stop_words = ['Austria', 'Austrian', 'Belgium', 'Belgian', 'Bulgaria', 'Bulgarian', 'Czech', 'Cyprus', 'Cypriot', 'Germany', 'German',
                                 'Denmark', 'Danish', 'Estonia', 'Estonian', 'Croatia', 'Croatian', 'Finland', 'Finnish', 'France', 'French', 'Malta', 'Maltese',
                                 'Luxembourg', 'Lithuania', 'Lithuanian', 'Latvia', 'Latvian', 'Italy', 'Italian', 'Ireland', 'Irish', 'Hungary', 'Hungarian',
                                 'Greece', 'Greek', 'Spain', 'Spanish', 'Netherlands', 'Dutch', 'Poland', 'Polish', 'Portugal', 'Portuguese', 'Romania', 'Romanian',
                                 'Sweden', 'Swedish', 'Slovenia', 'Slovenian', 'Slovakia', 'Slovak']

         extra_stop_words =  ['energy', 'figure', 'table', 'plan', "necp", 'national', 'use', "measure", "sector", "climate",
                              "plan", "dimension", "integrated", "section", "republic", "measures", "policies", "target", "objective", "policy",
                              "projection", "assessment", "federal", "government"]

         necp_processed["necp_lemmas"] = necp_docs.swifter.apply(lambda doc: [token.lemma_ for token in doc
                                                                              if not token.is_stop
                                                                              if not token.is_punct
                                                                              if not (token.lemma_ in countries_stop_words)
                                                                              if not (token.lemma_.lower() in extra_stop_words)
                                                                              if token.is_alpha])
```

```
Pandas Apply: 100%                                   453/453 [00:03<00:00, 123.84it/s]
```

```
In [23]: from gensim.models import Phrases
         bigram = Phrases(necp_processed["necp_lemmas"], min_count=20)
         for idx in necp_processed["necp_lemmas"].index:
             for token in bigram[necp_processed["necp_lemmas"][idx]]:
                 if '_' in token:
                     necp_processed["necp_lemmas"][idx].append(token)
```

```
/usr/local/lib/python3.7/dist-packages/gensim/models/phrases.py:598: UserWarning:

For a faster implementation, use the gensim.models.phrases.Phraser class
```

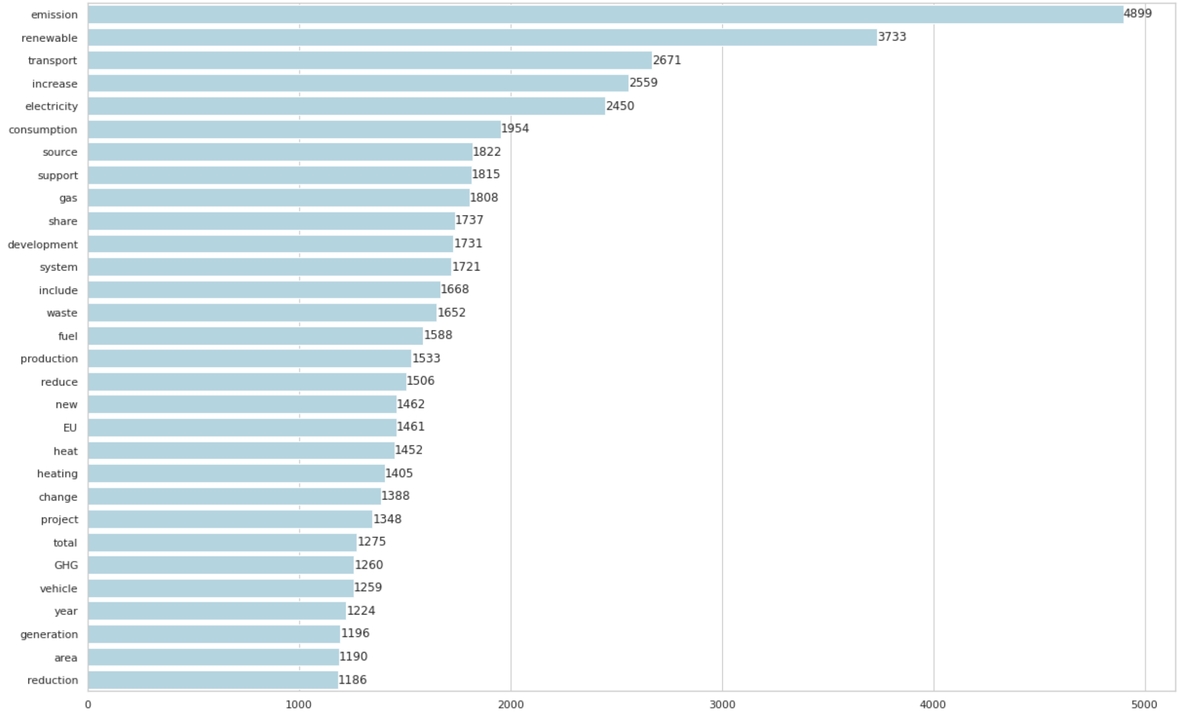
```
In [24]: def plot_counter(counter: Counter, orient: str = 'h', color: str='lightblue', figsize: tuple=(20,13)):
           plt.figure(figsize=figsize)
           keys = [k[0] for k in counter]
           vals = [int(k[1]) for k in counter]
           ax = sns.barplot(x=vals, y=keys, orient=orient, color=color)
           ax.bar_label(ax.containers[0])
           return ax
```

```
In [25]: from gensim.models import CoherenceModel
```

```
In [ ]:
```

Dimension: Decarbonisation

```
In [26]: decarbonisation_docs = necp_processed[(necp_processed['energy_union_dimension'] == "Decarbonisation")]["necp_lemmas"]
         decarbonisation_counter = Counter(decarbonisation_docs.sum()).most_common(30)
         plot_counter(decarbonisation_counter)
         plt.show()
```



```
In [27]: decarbonisation_docs = decarbonisation_docs.apply(lambda doc: [lemma for lemma in doc if not (lemma in ['emission', 'renewable'])])
```

In [28]:
```
decarbonisation_dictionary = Dictionary(decarbonisation_docs)
decarbonisation_dictionary.filter_extremes(no_below=2, no_above=1.0)
decarbonisation_encoded_docs = decarbonisation_docs.apply(decarbonisation_dictionary.doc2bow)
```

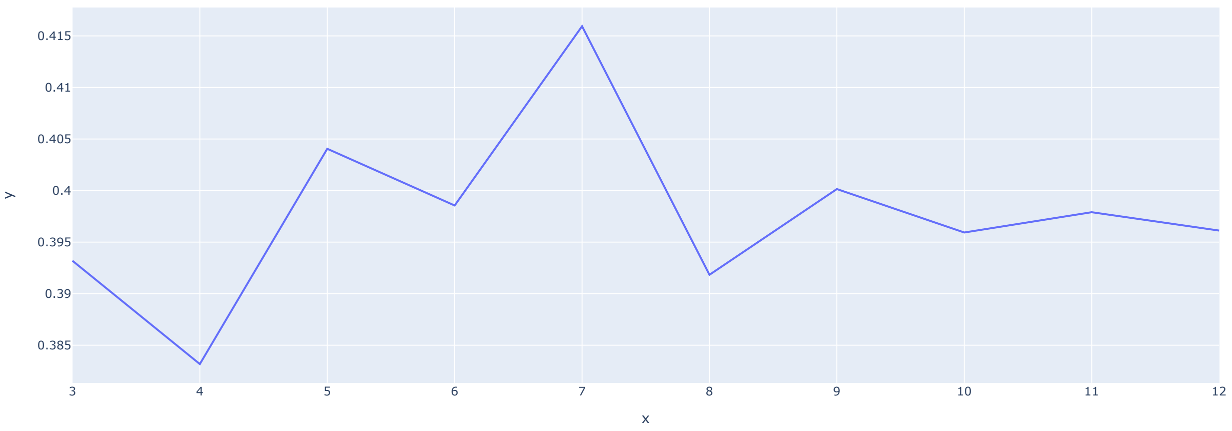
In [29]:
```
decarbonisation_models = []
for topics_number in tqdm(range(3, 13)):
    lda = LdaMulticore(decarbonisation_encoded_docs, num_topics=topics_number, passes=8, iterations=100, random_state=123)
    decarbonisation_models.append(lda)
```

```
  0%|          | 0/10 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning:

Calling np.sum(generator) is deprecated, and in the future will give a different result. Use np.sum(np.fromiter(generator)) or the python sum builtin instead.

100%|██████████| 10/10 [01:51<00:00, 11.11s/it]
```

In [30]:
```
decarbonisation_cvs = []
for model in tqdm(decarbonisation_models):
    cm = CoherenceModel(model,texts=decarbonisation_docs, dictionary=decarbonisation_dictionary)
    c_v = cm.get_coherence()
    decarbonisation_cvs.append(c_v)
```

```
100%|██████████| 10/10 [01:20<00:00,  8.03s/it]
```

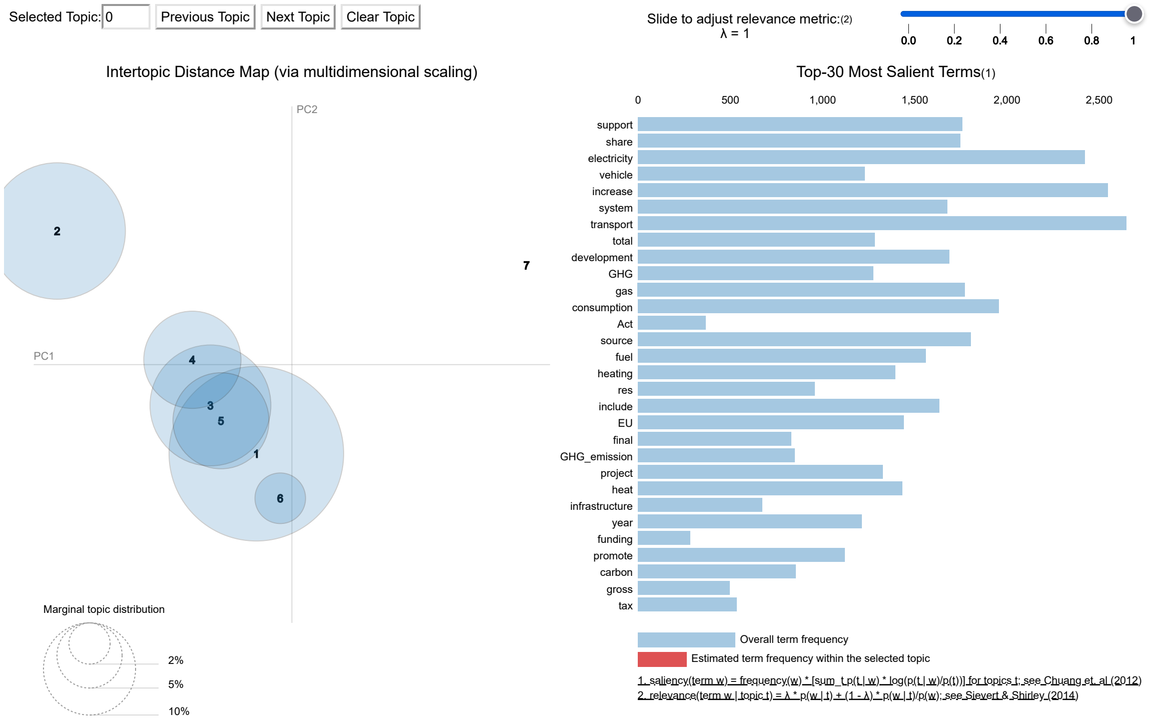
In [31]:
```
px.line(x=range(3, 13), y=decarbonisation_cvs)
```



In [32]:
```
vis = pyLDAvis.gensim_models.prepare(decarbonisation_models[4], decarbonisation_encoded_docs, dictionary=decarbonisation_dictionary)
vis
```

```
/usr/local/lib/python3.7/dist-packages/pyLDAvis/_prepare.py:247: FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
```

Out[32]:



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

```
In [33]: for idx, topic in decarbonisation_models[4].show_topics(formatted=False, num_words=15):
             print('Topic: {} \nWords: {}'.format(idx, [decarbonisation_dictionary[int(w[0])] for w in topic]))
```

```
Topic: 0
Words: ['gas', 'source', 'support', 'electricity', 'development', 'transport', 'greenhouse', 'increase', 'waste', 'include', 'land', 'fuel', 'system', 'reduction', 'produ
on']
Topic: 1
Words: ['res', 'include', 'development', 'increase', 'new', 'generation', 'electricity', 'system', 'support', 'vehicle', 'set', 'project', 'transport', 'action', 'share']
Topic: 2
Words: ['share', 'increase', 'consumption', 'electricity', 'total', 'transport', 'GHG', 'source', 'GHG_emission', 'final', 'heating', 'scenario', 'period', 'gas', 'year']
Topic: 3
Words: ['electricity', 'transport', 'vehicle', 'increase', 'EU', 'project', 'development', 'support', 'promote', 'heat', 'include', 'heating', 'fuel', 'source', 'gas']
Topic: 4
Words: ['transport', 'promote', 'support', 'reduce', 'system', 'increase', 'public', 'electricity', 'vehicle', 'new', 'change', 'development', 'area', 'include', 'fuel']
Topic: 5
Words: ['Act', 'system', 'transport', 'electricity', 'vehicle', 'support', 'funding', 'increase', 'fuel', 'development', 'Renewable', 'expansion', 'Sources', 'year', 'Ren
ble_Sources']
Topic: 6
Words: ['heat', 'support', 'production', 'system', 'electricity', 'waste', 'development', 'heating', 'fuel', 'include', 'transport', 'new', 'gas', 'increase', 'EU']
```

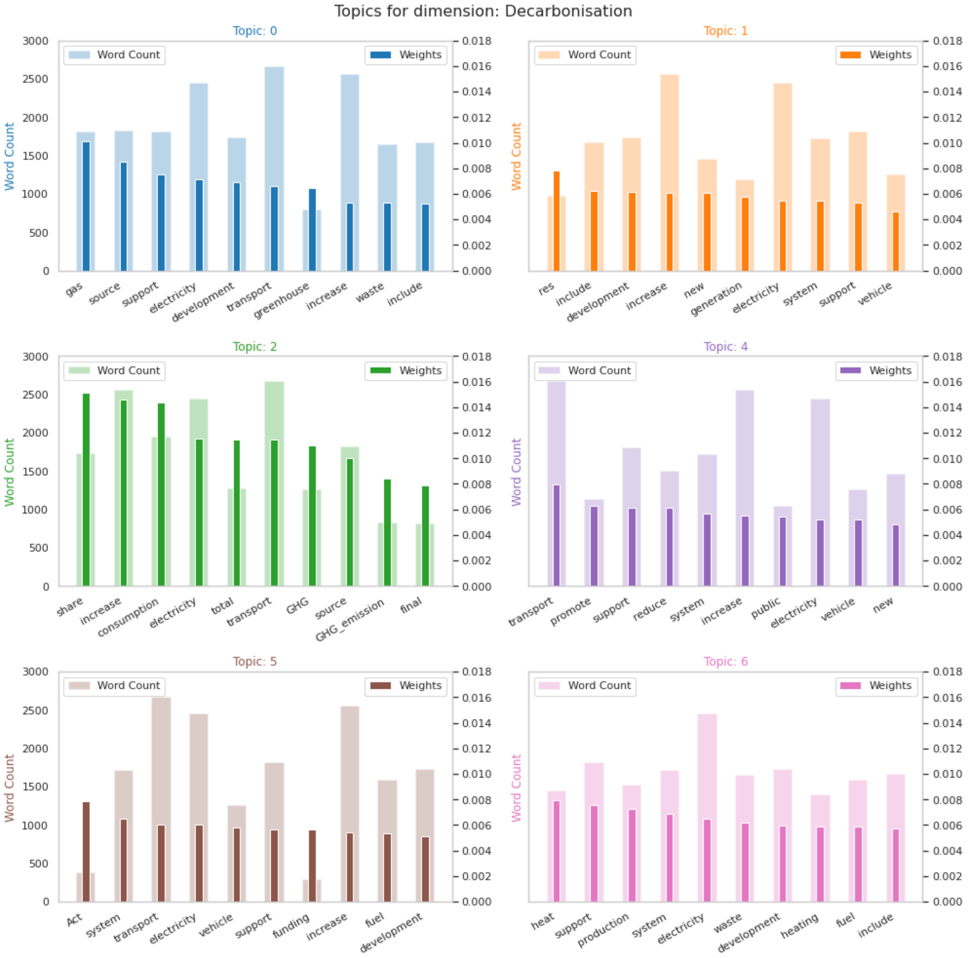
- Topic 0: gas, source, support, fuel, forest land
- Topic 1: renewable energy sources, development, new, power generation, electricity
- Topic 2: share, increase, consumption, total, GHG, gross, decrease

> 'The Commission envisions the EU as the global hub for developing next-generation renewable energies. It aims to make the EU the world leader in the sector through preparing markets and grids for a growing proportion of renewable energy, and investing in advanced, sustainable alternative fuels.'

- Topic 3: electricity, transport, vehicle -- do wyrzucenia (0% tokenów)
- Topic 4: transport, promote, support, public, system, vehicle, fuel, tax, mobility, encourage
- Topic 5: Act, system, funding, expansion (grid expansion)
- Topic 6: heat, waste, gas, district heating, biomass

In [34]:
```
from matplotlib import colors
topics = decarbonisation_models[4].show_topics(formatted=False)
counter = Counter(decarbonisation_docs.sum())

out = []
for i, topic in topics:
    for word, weight in topic:
        word = decarbonisation_dictionary[int(word)]
        out.append([word, i , weight, counter[word]])

df = pd.DataFrame(out, columns=['word', 'topic_id', 'importance', 'word_count'])

fig, axes = plt.subplots(3, 2, figsize=(14,14), sharey=True)
cols = [color for name, color in colors.TABLEAU_COLORS.items()]
for i, ax in enumerate(axes.flatten()):
    if i>=3:
      i+=1
    ax.bar(x='word', height="word_count", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.5, alpha=0.3, label='Word Count')
    ax_twin = ax.twinx()
    ax_twin.bar(x='word', height="importance", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.2, label='Weights')
    ax.set_ylabel('Word Count', color=cols[i])
    ax_twin.set_ylim(0, 0.018); ax.set_ylim(0, 3000)
    ax.set_title('Topic: ' + str(i), color=cols[i], fontsize=12)
    ax.tick_params(axis='y', left=False)
    ax.set_xticklabels(df.loc[df.topic_id==i, 'word'], rotation=30, horizontalalignment= 'right')
    ax.legend(loc='upper left'); ax_twin.legend(loc='upper right')
    ax.grid(False)
    ax_twin.grid(False)
fig.suptitle('Topics for dimension: Decarbonisation', fontsize=16)
fig.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: UserWarning:

FixedFormatter should only be used together with FixedLocator
```



In [35]:
```
decarbonisation_corpus_model = decarbonisation_models[4][decarbonisation_encoded_docs]
```

In [36]:
```
decarbonisation_metainfo = necp_processed[(necp_processed['energy_union_dimension'] == "Decarbonisation")]
res_len = len(decarbonisation_metainfo)
res = np.zeros((res_len, 7))
```

In [37]:
```
for i, doc in enumerate(decarbonisation_corpus_model):
  for topic in doc:
    res[i][topic[0]] = np.round(topic[1], 4)
```

In [38]:
```
decarbonisation_modeling_results = pd.concat([decarbonisation_metainfo.reset_index(drop=True), pd.DataFrame(res)], axis=1)
decarbonisation_topic_probs = decarbonisation_modeling_results.groupby("country").mean().loc[:,[0, 1, 2, 4, 5, 6]]
```

In [39]:

```
decarbonisation_modeling_results.groupby("subsection").mean().loc[:,[0, 1, 2, 4, 5, 6]]
```

Out[39]:

	0	1	2	4	5	6
subsection						
Current Situation and Reference Projections	0.063173	0.009588	0.868419	0.013788	0.006938	0.037131
National Objectives and Targets	0.115322	0.156689	0.532856	0.117607	0.012626	0.063419
Policies and Measures	0.115992	0.104942	0.012562	0.463119	0.039592	0.262388

In [40]:

```
decarbonisation_topic_probs
```

Out[40]:

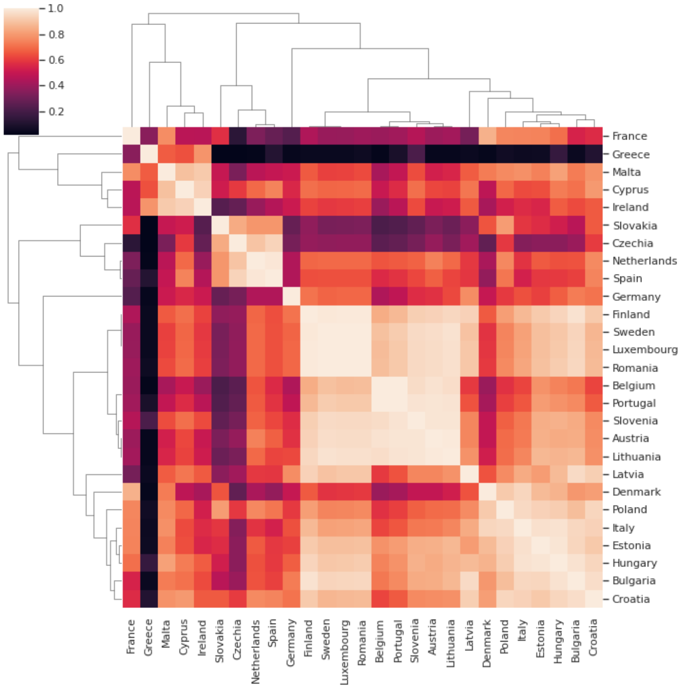
	0	1	2	4	5	6
country						
Austria	0.052133	0.000000	0.488200	0.447133	0.009833	0.000000
Belgium	0.000000	0.000000	0.410967	0.583767	0.000000	0.005067
Bulgaria	0.000000	0.000000	0.683567	0.152400	0.000000	0.162133
Croatia	0.147533	0.040267	0.576667	0.064000	0.000000	0.170233
Cyprus	0.184067	0.342767	0.403900	0.059167	0.000000	0.009833
Czechia	0.679300	0.000000	0.306233	0.011600	0.000000	0.000000
Denmark	0.000000	0.000000	0.448867	0.015733	0.000000	0.532800
Estonia	0.018167	0.010200	0.467767	0.230633	0.000000	0.273000
Finland	0.000000	0.000000	0.670700	0.273733	0.000000	0.053167
France	0.000000	0.195800	0.172800	0.148000	0.000000	0.483233
Germany	0.000000	0.000000	0.544833	0.000000	0.454833	0.000000
Greece	0.000000	0.955700	0.043850	0.000000	0.000000	0.000000
Hungary	0.000000	0.069700	0.522100	0.179033	0.000000	0.228767
Ireland	0.000000	0.524567	0.459367	0.013833	0.000000	0.000000
Italy	0.000000	0.006333	0.542267	0.107767	0.000000	0.343300
Latvia	0.000000	0.000000	0.974300	0.025050	0.000000	0.000000
Lithuania	0.000000	0.000000	0.510500	0.429900	0.042867	0.012367
Luxembourg	0.000000	0.000000	0.665700	0.333300	0.000000	0.000000
Malta	0.029933	0.353033	0.368500	0.029100	0.000000	0.219300
Netherlands	0.411033	0.000000	0.332767	0.219733	0.000000	0.035100
Poland	0.165267	0.014533	0.415267	0.090867	0.009367	0.300767
Portugal	0.000000	0.036533	0.429200	0.533967	0.000000	0.000000
Romania	0.000000	0.008800	0.653200	0.327333	0.000000	0.010067
Slovakia	0.455300	0.000000	0.217033	0.000000	0.000000	0.327467
Slovenia	0.000000	0.113033	0.469633	0.417000	0.000000	0.000000
Spain	0.447933	0.050100	0.340467	0.160933	0.000000	0.000000
Sweden	0.000000	0.000000	0.651600	0.346033	0.000000	0.000000

In [41]:

```
import scipy.spatial as sp
import scipy.cluster.hierarchy as hc
linkage = hc.linkage(decarbonisation_topic_probs, method='average', metric='cosine')
decarbonisation_similarities = sp.distance.squareform(sp.distance.pdist(decarbonisation_topic_probs.values, metric='cosine'))
```

In [42]:
```
plt.figure(figsize=(12, 8))
sns.clustermap(1-decarbonisation_similarities,
            xticklabels=decarbonisation_topic_probs.index,
            yticklabels=decarbonisation_topic_probs.index,
             row_linkage=linkage, col_linkage=linkage)
plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```



In [43]:
```
decarbonisation_comparison = decarbonisation_modeling_results.groupby(["country", "subsection"]).mean().loc[:,0:6]
```

In [44]:
```
countries = decarbonisation_modeling_results.country.unique()
sections = ["Policies and Measures", "National Objectives and Targets"]
```

In [45]:
```
decarbonisation_change = {"country": [], "similarity": []}
for country in countries:
  pm = decarbonisation_modeling_results.loc[(decarbonisation_modeling_results["country"] == country) &
                                        (decarbonisation_modeling_results["subsection"] == sections[0])].loc[:,0:6]
  noat = decarbonisation_modeling_results.loc[(decarbonisation_modeling_results["country"] == country) &
                                        (decarbonisation_modeling_results["subsection"] == sections[1])].loc[:,0:6]
  if pm.shape[0]==1:
    decarbonisation_change["country"].append(country)
    decarbonisation_change["similarity"].append(1-sp.distance.cosine(pm, noat))
pd.DataFrame(decarbonisation_change)
```

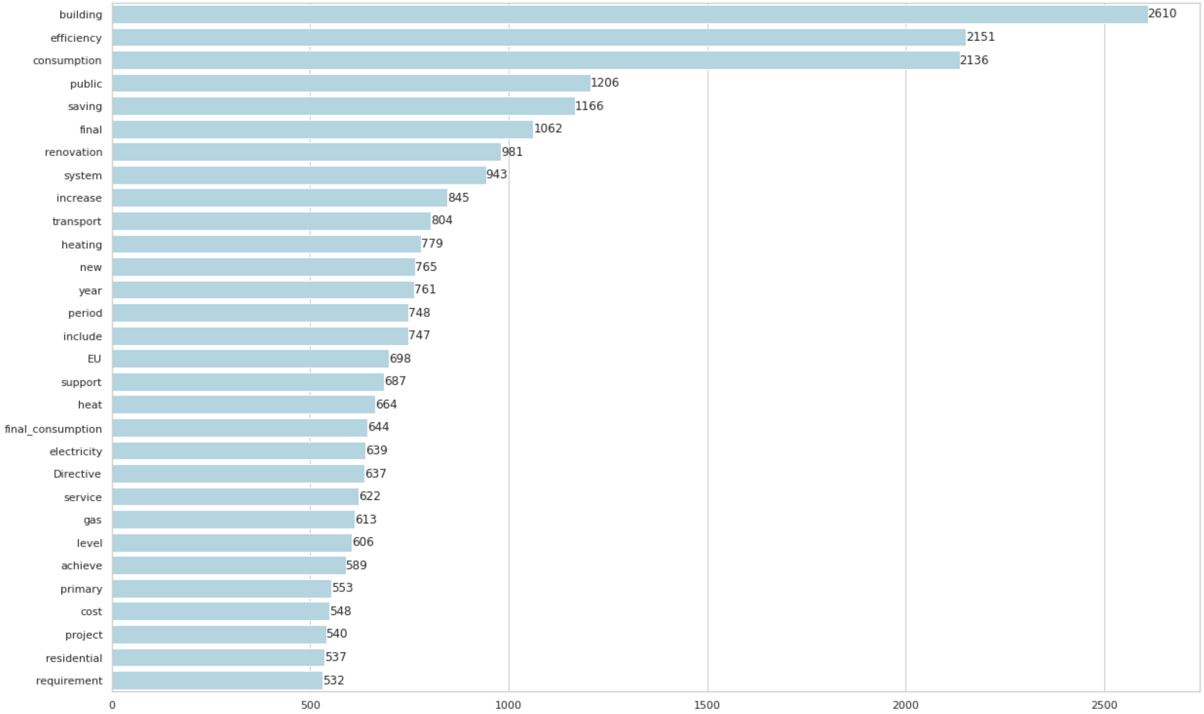
Out[45]:

	country	similarity
0	Austria	0.600718
1	Belgium	0.765183
2	Bulgaria	0.076474
3	Czechia	0.830240
4	Cyprus	0.421018
5	Germany	0.414432
6	Denmark	0.862707
7	Estonia	0.504848
8	Croatia	0.115934
9	Finland	0.338053
10	France	0.338781
11	Malta	0.353513
12	Luxembourg	0.000000
13	Lithuania	0.411048
14	Italy	0.385026
15	Ireland	0.847367
16	Hungary	0.243710
17	Greece	0.995409
18	Spain	0.913235
19	Netherlands	0.838799
20	Poland	0.298862
21	Portugal	0.875369
22	Romania	0.118285
23	Sweden	0.045384
24	Slovenia	0.306196
25	Slovakia	0.011229

In [ ]:

Dimension: Energy efficiency

```
In [46]: energy_efficiency_docs = necp_processed[(necp_processed['energy_union_dimension'] == "Energy efficiency")]["necp_lemmas"]
         energy_efficiency_counter = Counter(energy_efficiency_docs.sum()).most_common(30)
         plot_counter(energy_efficiency_counter)
         plt.show()
```



```
In [47]: energy_efficiency_docs = energy_efficiency_docs.apply(lambda doc: [lemma for lemma in doc if not (lemma in ['building', 'efficiency', 'consumption'])])
```

```
In [48]: energy_efficiency_dictionary = Dictionary(energy_efficiency_docs)
         energy_efficiency_dictionary.filter_extremes(no_below=2, no_above=1.0)
         energy_efficiency_encoded_docs = energy_efficiency_docs.apply(energy_efficiency_dictionary.doc2bow)
```

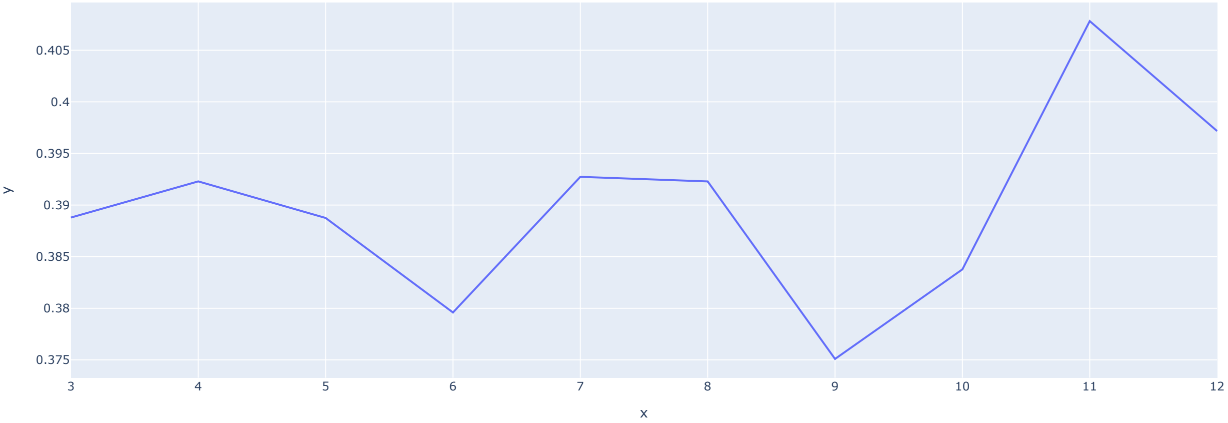
```
In [49]: energy_efficiency_models = []
         for topics_number in tqdm(range(3, 13)):
             lda = LdaMulticore(energy_efficiency_encoded_docs, num_topics=topics_number, passes=8, iterations=100, random_state=123)
             energy_efficiency_models.append(lda)
```

```
  0%|          | 0/10 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning:

Calling np.sum(generator) is deprecated, and in the future will give a different result. Use np.sum(np.fromiter(generator)) or the python sum builtin instead.

100%|██████████| 10/10 [01:06<00:00,  6.61s/it]
```

```
In [50]: energy_efficiency_cvs = []
         for model in tqdm(energy_efficiency_models):
             cm = CoherenceModel(model,texts=energy_efficiency_docs, dictionary=energy_efficiency_dictionary)
             c_v = cm.get_coherence()
             energy_efficiency_cvs.append(c_v)
```

```
100%|██████████| 10/10 [00:47<00:00,  4.73s/it]
```

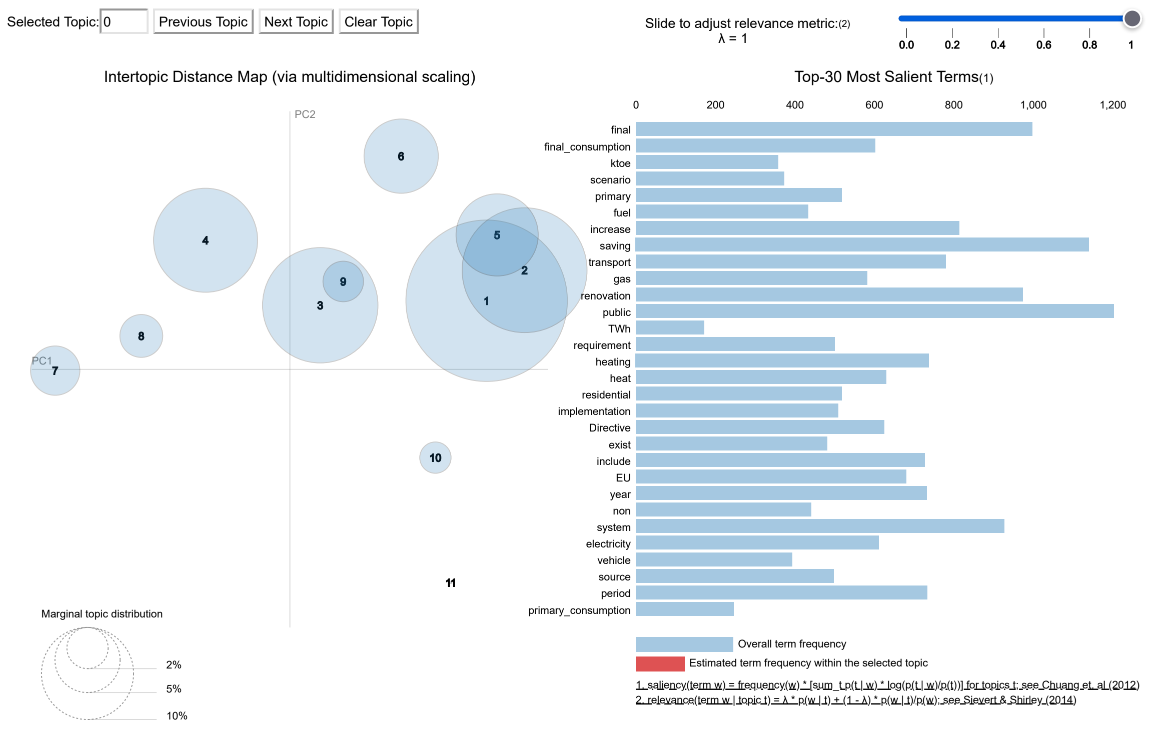
```
In [51]: px.line(x=range(3, 13), y=energy_efficiency_cvs)
```



```
In [52]: vis = pyLDAvis.gensim_models.prepare(energy_efficiency_models[8], energy_efficiency_encoded_docs, dictionary=energy_efficiency_dictionary)
         vis
```

```
/usr/local/lib/python3.7/dist-packages/pyLDAvis/_prepare.py:247: FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
```

Out[52]:



```
In [53]: for idx, topic in energy_efficiency_models[8].show_topics(formatted=False, num_words=15, num_topics=11):
             print('Topic: {} \nWords: {}'.format(idx, [energy_efficiency_dictionary[int(w[0])] for w in topic]))
```

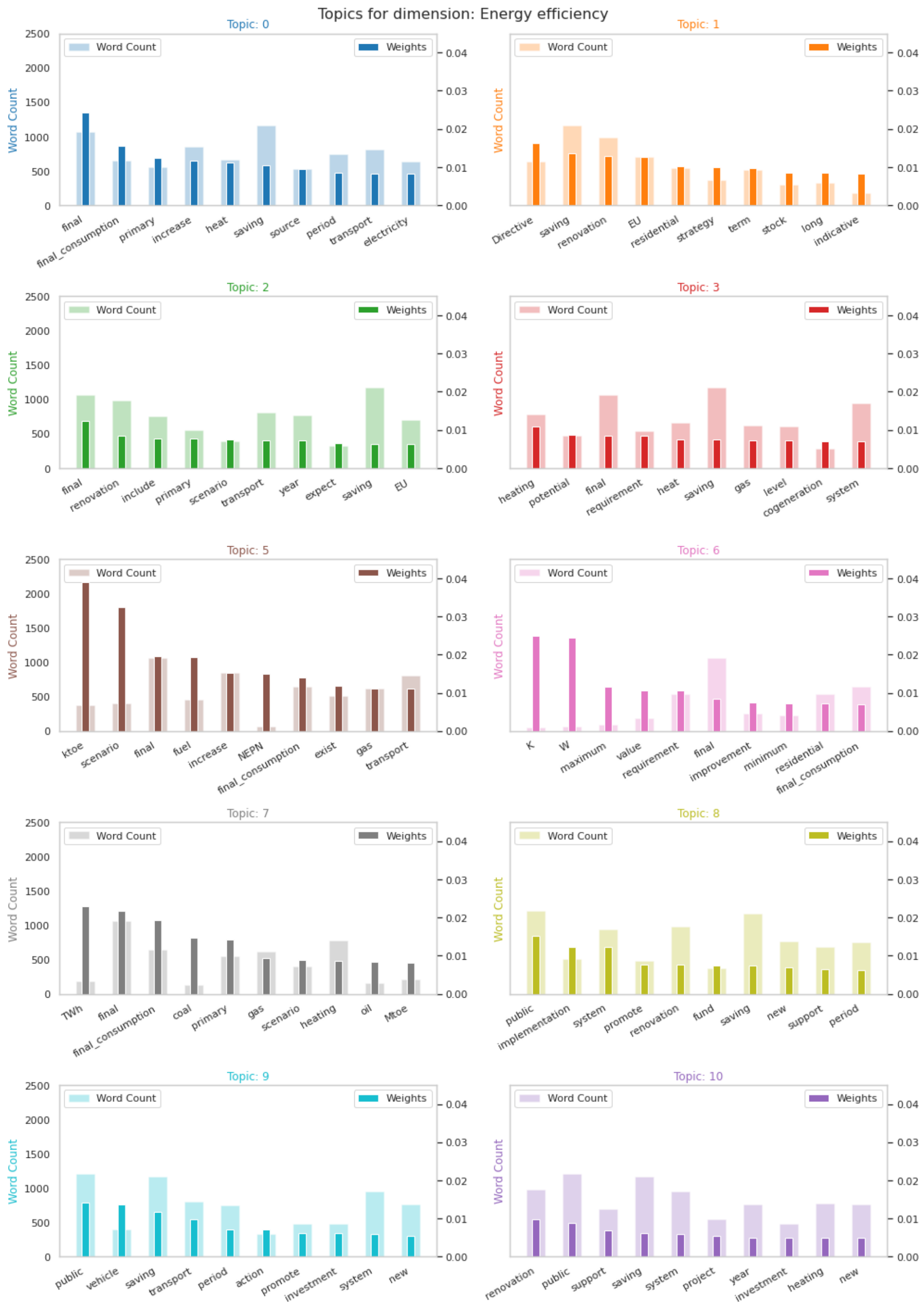
```
Topic: 0
Words: ['final', 'final_consumption', 'primary', 'increase', 'heat', 'saving', 'source', 'period', 'transport', 'electricity', 'EU', 'industry', 'total', 'year', 'level']
Topic: 1
Words: ['Directive', 'saving', 'renovation', 'EU', 'residential', 'strategy', 'term', 'stock', 'long', 'indicative', 'long_term', 'set', 'include', 'period', 'contributio
Topic: 2
Words: ['final', 'renovation', 'include', 'primary', 'scenario', 'transport', 'year', 'expect', 'saving', 'EU', 'Mtoe', 'vehicle', 'cost', 'PPM', 'final_consumption']
Topic: 3
Words: ['heating', 'potential', 'final', 'requirement', 'heat', 'saving', 'gas', 'level', 'cogeneration', 'system', 'year', 'cost', 'electricity', 'high', 'increase']
Topic: 4
Words: ['public', 'saving', 'final', 'system', 'transport', 'period', 'service', 'renovation', 'increase', 'gas', 'support', 'new', 'promote', 'electricity', 'final_consu
ion']
Topic: 5
Words: ['ktoe', 'scenario', 'final', 'fuel', 'increase', 'NEPN', 'final_consumption', 'exist', 'gas', 'transport', 'non', 'NEPN_scenario', 'Scenario', 'electricity', 'nat
l']
Topic: 6
Words: ['K', 'W', 'maximum', 'value', 'requirement', 'final', 'improvement', 'minimum', 'residential', 'final_consumption', 'u', 'saving', 'efficient', 'agreement', 'prim
']
Topic: 7
Words: ['TWh', 'final', 'final_consumption', 'coal', 'primary', 'gas', 'scenario', 'heating', 'oil', 'Mtoe', 'twh', 'primary_consumption', 'transport', 'industry', 'fuel'
Topic: 8
Words: ['public', 'implementation', 'system', 'promote', 'renovation', 'fund', 'saving', 'new', 'support', 'period', 'service', 'project', 'transport', 'implement', 'fina
al']
Topic: 9
Words: ['public', 'vehicle', 'saving', 'transport', 'period', 'action', 'promote', 'investment', 'system', 'new', 'include', 'aim', 'million', 'improve', 'achieve']
Topic: 10
Words: ['renovation', 'public', 'support', 'saving', 'system', 'project', 'year', 'investment', 'heating', 'new', 'service', 'performance', 'term', 'programme', 'order']
```

- Topic 0: final, final_consumption, primary, saving
- Topic 1: Directive, saving, renovation, strategy, long
- Topic 2: PPM, PPM scenario, power generation, scenario, final consumption
- Topic 3: heating, heat, potential, final, requirement
- Topic 4: public, saving, final -- do wyrzucenia (0% tokenów)
- Topic 5: ktoe (kilotonne of oil equivalent), scenario, fuel, baseline
- Topic 6: K, W, maximum, EEOS (Energy Efficiency Obligation Scheme)
- Topic 7: TWh, coal, gas, oil, Consommation
- Topic 8: public, implementation, system, instrument
- Topic 9: public, vehicle, autonomous, autonomous community
- Topic 10: renovation, support, work, founding

In [54]:

```
from matplotlib import colors
topics = energy_efficiency_models[8].show_topics(formatted=False, num_topics=11)
counter = Counter(energy_efficiency_docs.sum())

out = []
for i, topic in topics:
    for word, weight in topic:
        word = energy_efficiency_dictionary[int(word)]
        out.append([word, i , weight, counter[word]])

df = pd.DataFrame(out, columns=['word', 'topic_id', 'importance', 'word_count'])

fig, axes = plt.subplots(5, 2, figsize=(14, 20), sharey=True)
cols = [color for name, color in colors.TABLEAU_COLORS.items()]
cols.append(cols[4])

for i, ax in enumerate(axes.flatten()):
    if i>=4:
      i+=1
    ax.bar(x='word', height="word_count", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.5, alpha=0.3, label='Word Count')
    ax_twin = ax.twinx()
    ax_twin.bar(x='word', height="importance", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.2, label='Weights')
    ax.set_ylabel('Word Count', color=cols[i])
    ax_twin.set_ylim(0, 0.045);
    ax.set_ylim(0, 2500)
    ax.set_title('Topic: ' + str(i), color=cols[i], fontsize=12)
    ax.tick_params(axis='y', left=False)
    ax.set_xticklabels(df.loc[df.topic_id==i, 'word'], rotation=30, horizontalalignment= 'right')
    ax.legend(loc='upper left'); ax_twin.legend(loc='upper right')
    ax.grid(False)
    ax_twin.grid(False)
fig.suptitle('Topics for dimension: Energy efficiency', fontsize=16)
fig.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:28: UserWarning:

FixedFormatter should only be used together with FixedLocator
```



In [55]:
```
energy_efficiency_corpus_model = energy_efficiency_models[8][energy_efficiency_encoded_docs]
```

In [56]:
```
energy_efficiency_metainfo = necp_processed[(necp_processed['energy_union_dimension'] == "Energy efficiency")]
res_len = len(energy_efficiency_metainfo)
res = np.zeros((res_len, 11))
```

In [57]:
```
for i, doc in enumerate(energy_efficiency_corpus_model):
  for topic in doc:
    res[i][topic[0]] = np.round(topic[1], 4)
```

In [58]:
```
energy_efficiency_modeling_results = pd.concat([energy_efficiency_metainfo.reset_index(drop=True), pd.DataFrame(res)], axis=1)
energy_efficiency_topic_probs = energy_efficiency_modeling_results.groupby("country").mean().loc[:,[0, 1, 2, 4, 5, 6, 7, 8, 9, 10]]
```

In [59]:
```
energy_efficiency_topic_probs
```

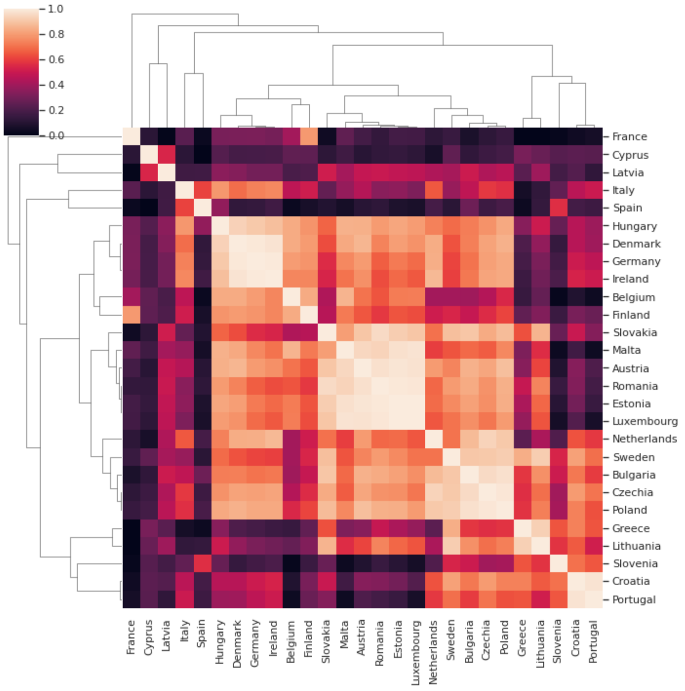
Out[59]:

	0	1	2	4	5	6	7	8	9	10
country										
Austria	0.581100	0.155200	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.260967
Belgium	0.186000	0.009767	0.000000	0.0	0.000000	0.000000	0.011200	0.000000	0.000000	0.459600
Bulgaria	0.261433	0.164133	0.000000	0.0	0.073167	0.000000	0.000000	0.101400	0.000000	0.033433
Croatia	0.095400	0.313700	0.000000	0.0	0.033033	0.000000	0.000000	0.333300	0.000000	0.000000
Cyprus	0.000000	0.000000	0.347800	0.0	0.000000	0.332933	0.000000	0.175833	0.000000	0.142833
Czechia	0.303533	0.287100	0.000000	0.0	0.000000	0.000000	0.000000	0.147633	0.000000	0.087600
Denmark	0.261600	0.252867	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.333167
Estonia	0.438967	0.039733	0.000000	0.0	0.000000	0.000000	0.000000	0.040033	0.000000	0.202067
Finland	0.097967	0.062800	0.000000	0.0	0.000000	0.000000	0.149100	0.038033	0.000000	0.214767
France	0.000000	0.044167	0.000000	0.0	0.000000	0.000000	0.644967	0.000000	0.015867	0.294267
Germany	0.208433	0.286100	0.000000	0.0	0.000000	0.000000	0.000000	0.007967	0.000000	0.322700
Greece	0.275000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.658100	0.005050	0.000000
Hungary	0.213433	0.135900	0.000000	0.0	0.016533	0.000000	0.000000	0.050233	0.116800	0.253467
Ireland	0.224467	0.359033	0.016633	0.0	0.017033	0.000000	0.000000	0.000000	0.000000	0.340767
Italy	0.044767	0.207600	0.000000	0.0	0.007733	0.000000	0.000000	0.000000	0.200833	0.141233
Latvia	0.215800	0.049533	0.000000	0.0	0.063267	0.352000	0.000000	0.015567	0.010933	0.000000
Lithuania	0.193100	0.000000	0.013167	0.0	0.021767	0.000000	0.000000	0.201400	0.015467	0.000000
Luxembourg	0.661300	0.013933	0.000000	0.0	0.000000	0.000000	0.000000	0.019200	0.000000	0.304600
Malta	0.469000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.005633	0.000000	0.353867
Netherlands	0.385867	0.482433	0.000000	0.0	0.000000	0.018467	0.000000	0.000000	0.000000	0.108500
Poland	0.139433	0.102633	0.000000	0.0	0.011867	0.000000	0.000000	0.060300	0.000000	0.053067
Portugal	0.015000	0.332833	0.000000	0.0	0.000000	0.000000	0.004033	0.311100	0.017967	0.004133
Romania	0.601200	0.042133	0.000000	0.0	0.000000	0.000000	0.000000	0.087767	0.038533	0.202267
Slovakia	0.537433	0.080000	0.000000	0.0	0.000000	0.000000	0.000000	0.167800	0.000000	0.047100
Slovenia	0.020233	0.139000	0.000000	0.0	0.386400	0.000000	0.000000	0.386200	0.067700	0.000000
Spain	0.042733	0.108033	0.000000	0.0	0.374433	0.000000	0.000000	0.000000	0.453033	0.000000
Sweden	0.269333	0.133867	0.000000	0.0	0.000000	0.000000	0.028400	0.247333	0.000000	0.060467

In [60]:
```
import scipy.spatial as sp
import scipy.cluster.hierarchy as hc
linkage = hc.linkage(energy_efficiency_topic_probs, method='average', metric='cosine')
energy_efficiency_similarities = sp.distance.squareform(sp.distance.pdist(energy_efficiency_topic_probs.values, metric='cosine'))
```

In [61]:
```
plt.figure(figsize=(12, 8))
sns.clustermap(1-energy_efficiency_similarities,
           xticklabels=energy_efficiency_topic_probs.index,
           yticklabels=energy_efficiency_topic_probs.index,
            row_linkage=linkage, col_linkage=linkage)
plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```



In [62]:
```
energy_efficiency_comparison = energy_efficiency_modeling_results.groupby(["country", "subsection"]).mean().loc[:,0:10]
```

In [63]:
```
countries = energy_efficiency_modeling_results.country.unique()
sections = ["Policies and Measures", "National Objectives and Targets"]
```

In [64]:

```
energy_efficiency_change = {"country": [], "similarity": []}
for country in countries:
  pm = energy_efficiency_modeling_results.loc[(energy_efficiency_modeling_results["country"] == country) &
                                          (energy_efficiency_modeling_results["subsection"] == sections[0])].loc[:,0:10]
  noat = energy_efficiency_modeling_results.loc[(energy_efficiency_modeling_results["country"] == country) &
                                          (energy_efficiency_modeling_results["subsection"] == sections[1])].loc[:,0:10]
  if pm.shape[0]==1:
    energy_efficiency_change["country"].append(country)
    energy_efficiency_change["similarity"].append(1-sp.distance.cosine(pm, noat))
pd.DataFrame(energy_efficiency_change)
```

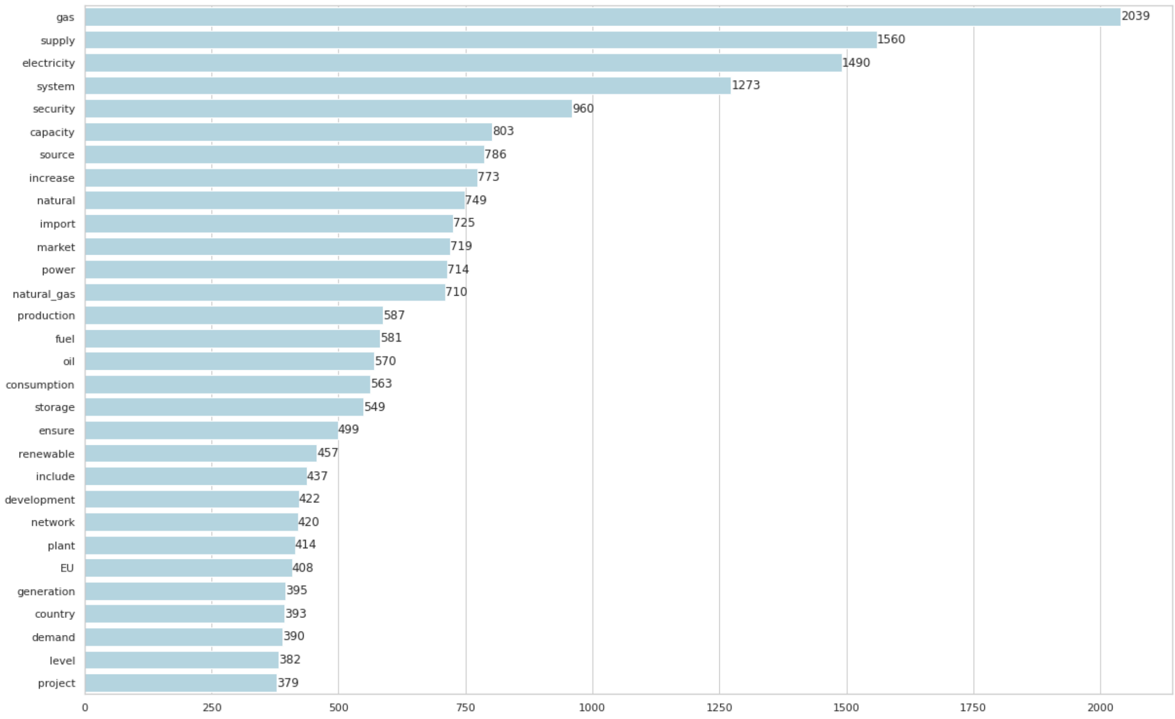
Out[64]:

	country	similarity
0	Austria	0.084216
1	Belgium	0.560438
2	Bulgaria	0.442405
3	Czechia	0.019445
4	Cyprus	0.064169
5	Germany	0.000000
6	Denmark	0.000000
7	Estonia	0.000000
8	Croatia	0.000000
9	Finland	0.180263
10	France	0.077137
11	Malta	0.207696
12	Luxembourg	0.013867
13	Lithuania	0.554290
14	Latvia	0.116160
15	Italy	0.030940
16	Ireland	0.146897
17	Hungary	0.090562
18	Greece	0.617696
19	Spain	0.692930
20	Netherlands	0.879926
21	Poland	0.821237
22	Portugal	0.000000
23	Romania	0.051030
24	Sweden	0.000000
25	Slovenia	0.306061
26	Slovakia	0.011046

Dimension: Energy security

In [65]:

```
energy_security_docs = necp_processed[(necp_processed['energy_union_dimension'] == "Energy security")]["necp_lemmas"]
energy_security_counter = Counter(energy_security_docs.sum()).most_common(30)
plot_counter(energy_security_counter)
plt.show()
```



In [66]:

```
energy_security_docs = energy_security_docs.apply(lambda doc: [lemma for lemma in doc if not (lemma in ['gas', 'supply', 'electricity', 'system', 'security
'])])
```

In [67]:
```
energy_security_dictionary = Dictionary(energy_security_docs)
energy_security_dictionary.filter_extremes(no_below=2, no_above=1.0)
energy_security_encoded_docs = energy_security_docs.apply(energy_security_dictionary.doc2bow)
```

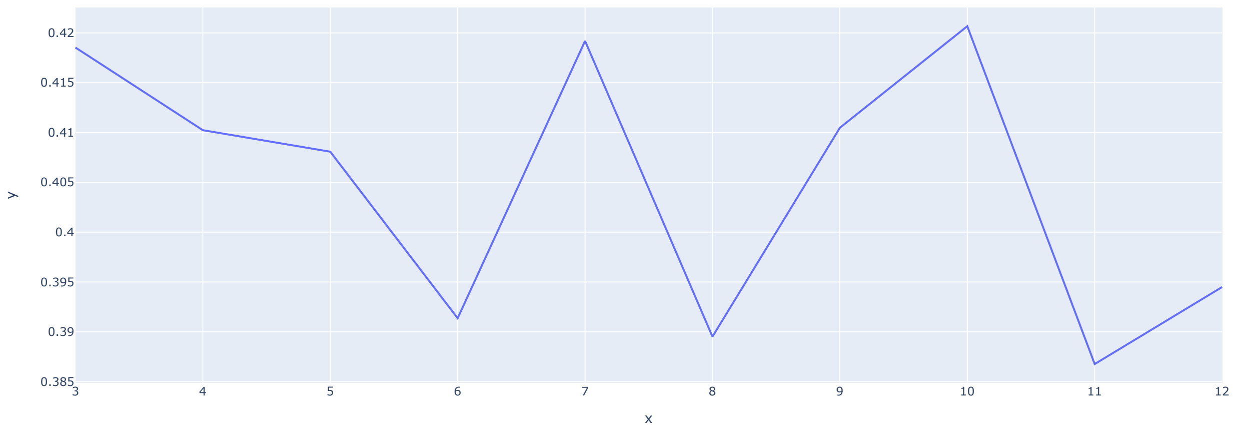
In [68]:
```
energy_security_models = []
for topics_number in tqdm(range(3, 13)):
    lda = LdaMulticore(energy_security_encoded_docs, num_topics=topics_number, passes=8, iterations=100, random_state=123)
    energy_security_models.append(lda)
```

```
  0%|          | 0/10 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning:

Calling np.sum(generator) is deprecated, and in the future will give a different result. Use np.sum(np.fromiter(generator)) or the python sum builtin instead.

100%|██████████| 10/10 [00:51<00:00,  5.15s/it]
```

In [69]:
```
energy_security_cvs = []
for model in tqdm(energy_security_models):
    cm = CoherenceModel(model,texts=energy_security_docs, dictionary=energy_security_dictionary)
    c_v = cm.get_coherence()
    energy_security_cvs.append(c_v)
```

```
100%|██████████| 10/10 [00:26<00:00,  2.66s/it]
```

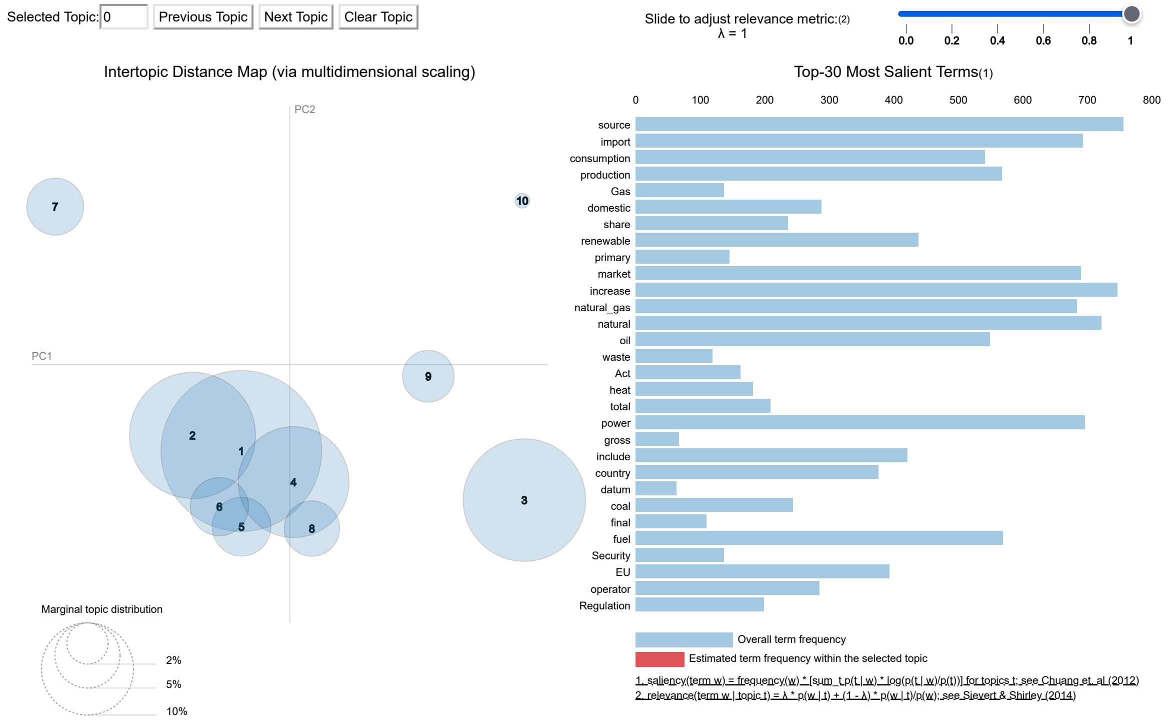
In [70]:
```
px.line(x=range(3, 13), y=energy_security_cvs)
```



In [71]:
```
vis = pyLDAvis.gensim_models.prepare(energy_security_models[7], energy_security_encoded_docs, dictionary=energy_security_dictionary)
vis
```

```
/usr/local/lib/python3.7/dist-packages/pyLDAvis/_prepare.py:247: FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
```

Out[71]:



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

In [72]:
```
for idx, topic in energy_security_models[7].show_topics(formatted=False, num_words=15):
    print('Topic: {} \nWords: {}'.format(idx, [energy_security_dictionary[int(w[0])] for w in topic]))
```

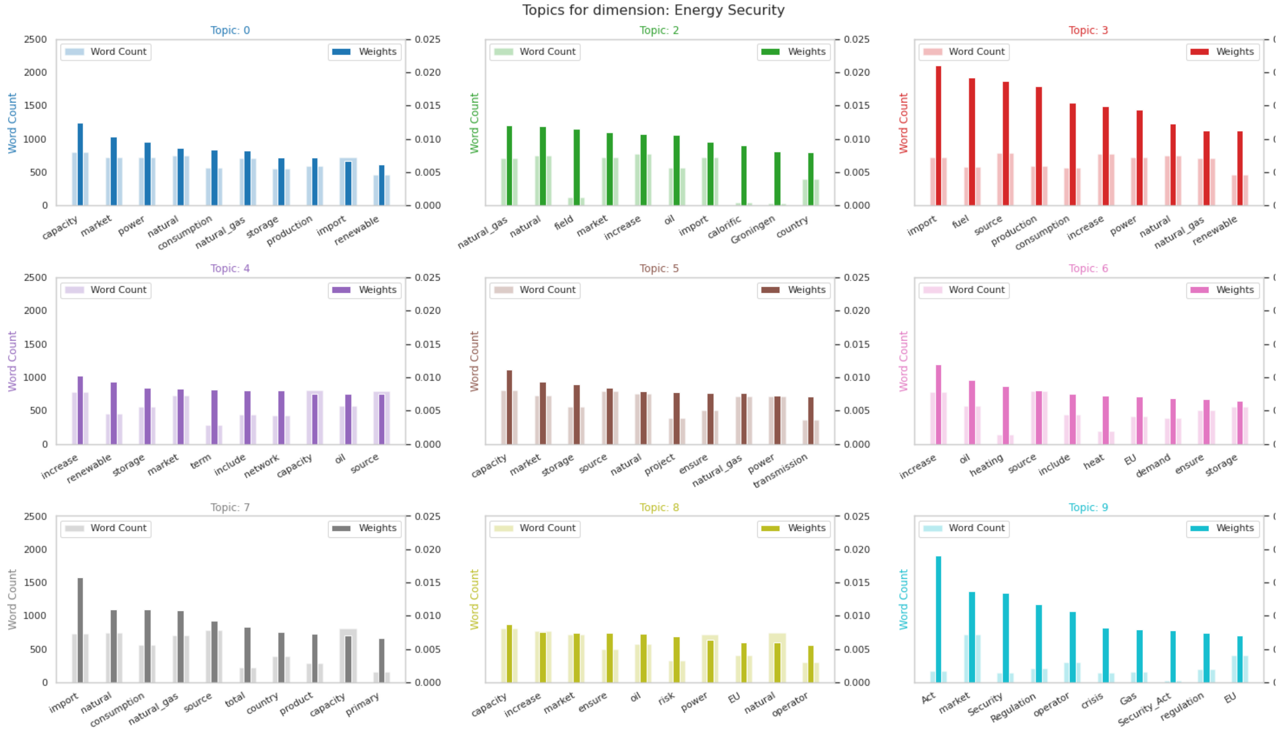
```
Topic: 0
Words: ['capacity', 'market', 'power', 'natural', 'consumption', 'natural_gas', 'storage', 'production', 'import', 'renewable', 'increase', 'ensure', 'term', 'nuclear', '
and']
Topic: 1
Words: ['Gas', 'source', 'import', 'consumption', 'domestic', 'datum', 'Oil', 'PJ', 'fact', 'production', 'gross', 'primary', 'waste', 'share', 'Wind']
Topic: 2
Words: ['natural_gas', 'natural', 'field', 'market', 'increase', 'oil', 'import', 'calorific', 'Groningen', 'country', 'demand', 'capacity', 'extraction', 'small', 'year'
Topic: 3
Words: ['import', 'fuel', 'source', 'production', 'consumption', 'increase', 'power', 'natural', 'natural_gas', 'renewable', 'plant', 'share', 'coal', 'oil', 'domestic']
Topic: 4
Words: ['increase', 'renewable', 'storage', 'market', 'term', 'include', 'network', 'capacity', 'oil', 'source', 'demand', 'EU', 'country', 'development', 'project']
Topic: 5
Words: ['capacity', 'market', 'storage', 'source', 'natural', 'project', 'ensure', 'natural_gas', 'power', 'transmission', 'network', 'increase', 'development', 'emergenc
'EU']
Topic: 6
Words: ['increase', 'oil', 'heating', 'source', 'include', 'heat', 'EU', 'demand', 'ensure', 'storage', 'renewable', 'continue', 'danish', 'import', 'risk']
Topic: 7
Words: ['import', 'natural', 'consumption', 'natural_gas', 'source', 'total', 'country', 'product', 'capacity', 'primary', 'renewable', 'coal', 'dependency', 'increase',
el']
Topic: 8
Words: ['capacity', 'increase', 'market', 'ensure', 'oil', 'risk', 'power', 'EU', 'natural', 'operator', 'demand', 'include', 'natural_gas', 'storage', 'regional']
Topic: 9
Words: ['Act', 'market', 'Security', 'Regulation', 'operator', 'crisis', 'Gas', 'Security_Act', 'regulation', 'EU', 'Supply', 'basis', 'Industry_Act', 'Industry', 'Electr
ty']
```

- Topic 0: wood, forest, crop, power, natural, natural gas, nuclear, demand
- Topic 1: Gas, source, import -- do wyrzucenia (0% tokenów)
- Topic 2: natural_gas, natural, field, oil
- Topic 3: fuel, import, production, share
- Topic 4: increase, renewable, oil, demand
- Topic 5: storage, capacity, emergency
- Topic 6: increase, oil, heating, Agreement
- Topic 7: import, natural, renewable, consumption
- Topic 8: risk, ensure, regional
- Topic 9: Act, Regulation, Security Act

In [73]:

```
from matplotlib import colors
topics = energy_security_models[7].show_topics(formatted=False)
counter = Counter(energy_security_docs.sum())

out = []
for i, topic in topics:
    for word, weight in topic:
        word = energy_security_dictionary[int(word)]
        out.append([word, i , weight, counter[word]])

df = pd.DataFrame(out, columns=['word', 'topic_id', 'importance', 'word_count'])

fig, axes = plt.subplots(3, 3, figsize=(21,12), sharey=True)
cols = [color for name, color in colors.TABLEAU_COLORS.items()]
for i, ax in enumerate(axes.flatten()):
    if i>=1:
      i+=1
    ax.bar(x='word', height="word_count", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.5, alpha=0.3, label='Word Count')
    ax_twin = ax.twinx()
    ax_twin.bar(x='word', height="importance", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.2, label='Weights')
    ax.set_ylabel('Word Count', color=cols[i])
    ax_twin.set_ylim(0, 0.025);
    ax.set_ylim(0, 2500)
    ax.set_title('Topic: ' + str(i), color=cols[i], fontsize=12)
    ax.tick_params(axis='y', left=False)
    ax.set_xticklabels(df.loc[df.topic_id==i, 'word'], rotation=30, horizontalalignment= 'right')
    ax.legend(loc='upper left'); ax_twin.legend(loc='upper right')
    ax.grid(False)
    ax_twin.grid(False)
fig.suptitle('Topics for dimension: Energy Security', fontsize=16)
fig.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:26: UserWarning:

FixedFormatter should only be used together with FixedLocator
```



In [74]:

```
energy_security_corpus_model = energy_security_models[7][energy_security_encoded_docs]
```

In [75]:

```
energy_security_metainfo = necp_processed[(necp_processed['energy_union_dimension'] == "Energy security")]
res_len = len(energy_security_metainfo)
res = np.zeros((res_len, 10))
```

In [76]:

```
for i, doc in enumerate(energy_security_corpus_model):
  for topic in doc:
    res[i][topic[0]] = np.round(topic[1], 4)
```

In [77]:

```
energy_security_modeling_results = pd.concat([energy_security_metainfo.reset_index(drop=True), pd.DataFrame(res)], axis=1)
energy_security_topic_probs = energy_security_modeling_results.groupby("country").mean().loc[:,[0, 1, 2, 4, 5, 6, 7, 8, 9]]
```

```
In [78]: energy_security_topic_probs
```

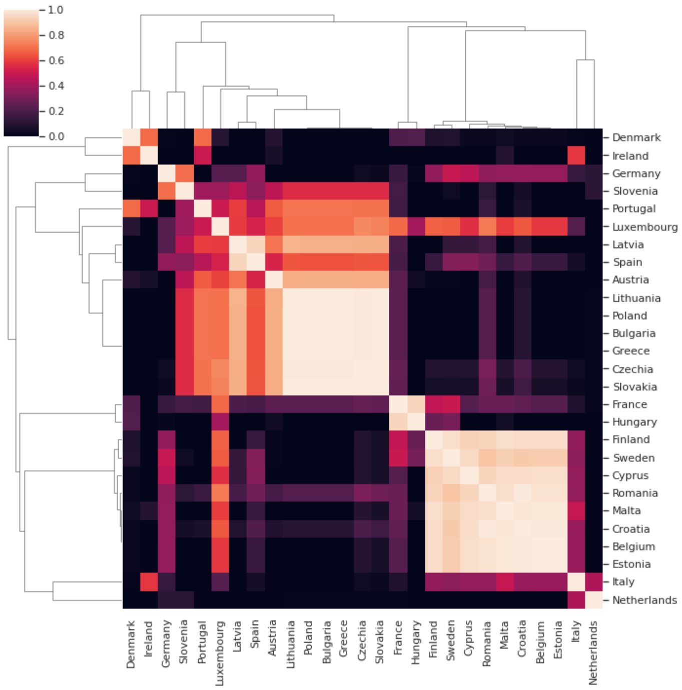
Out[78]:

	0	1	2	4	5	6	7	8	9
country									
Austria	0.028100	0.331967	0.000000	0.003600	0.522267	0.053567	0.000000	0.000000	0.000000
Belgium	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.665800	0.000000
Bulgaria	0.000000	0.000000	0.000000	0.000000	0.578700	0.000000	0.000000	0.000000	0.000000
Croatia	0.000000	0.000000	0.000000	0.000000	0.081333	0.000000	0.000000	0.791800	0.000000
Cyprus	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.165300	0.634267	0.000000
Czechia	0.000000	0.000000	0.000000	0.000000	0.663800	0.000000	0.000000	0.064067	0.000000
Denmark	0.102800	0.000000	0.000000	0.000000	0.000000	0.470833	0.000000	0.012067	0.000000
Estonia	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.665933	0.000000
Finland	0.160900	0.000000	0.000000	0.000000	0.000000	0.000000	0.004067	0.571000	0.000000
France	0.620400	0.000000	0.000000	0.000000	0.156267	0.000000	0.000000	0.148933	0.036467
Germany	0.008267	0.000000	0.000000	0.000000	0.000000	0.000000	0.201867	0.181600	0.406933
Greece	0.000000	0.000000	0.000000	0.000000	0.986000	0.000000	0.000000	0.000000	0.000000
Hungary	0.666400	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Ireland	0.000000	0.000000	0.000000	0.332967	0.000000	0.333000	0.000000	0.000000	0.000000
Italy	0.000000	0.000000	0.175867	0.333233	0.000000	0.000000	0.000000	0.154533	0.000000
Latvia	0.000000	0.000000	0.000000	0.000000	0.525233	0.000000	0.332933	0.000000	0.000000
Lithuania	0.000000	0.000000	0.000000	0.000000	0.773067	0.000000	0.011167	0.000000	0.000000
Luxembourg	0.219567	0.000000	0.000000	0.000000	0.377700	0.000000	0.000000	0.313433	0.000000
Malta	0.025067	0.000000	0.000000	0.066000	0.000000	0.000000	0.000000	0.491133	0.000000
Netherlands	0.000000	0.000000	0.866867	0.000000	0.006967	0.000000	0.000000	0.000000	0.117667
Poland	0.000000	0.000000	0.000000	0.000000	0.559000	0.000000	0.000000	0.000000	0.000000
Portugal	0.000000	0.000000	0.000000	0.000000	0.333133	0.332667	0.000000	0.000000	0.000000
Romania	0.000000	0.000000	0.000000	0.000000	0.106200	0.000000	0.000000	0.480467	0.000000
Slovakia	0.000000	0.000000	0.000000	0.000000	0.850067	0.000000	0.000000	0.054700	0.000000
Slovenia	0.000000	0.000000	0.000000	0.000000	0.219733	0.000000	0.000000	0.000000	0.330267
Spain	0.000000	0.000000	0.000000	0.000000	0.391800	0.000000	0.468233	0.090033	0.000000
Sweden	0.165133	0.000000	0.000000	0.000000	0.000000	0.000000	0.140633	0.500433	0.024733

```
In [79]: linkage = hc.linkage(energy_security_topic_probs, method='average', metric='cosine')
         energy_security_similarities = sp.distance.squareform(sp.distance.pdist(energy_security_topic_probs.values, metric='cosine'))
```

```
In [80]: plt.figure(figsize=(12, 8))
         sns.clustermap(1-energy_security_similarities,
                     xticklabels=energy_security_topic_probs.index,
                     yticklabels=energy_security_topic_probs.index,
                      row_linkage=linkage, col_linkage=linkage)
         plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```



```
In [81]: energy_security_comparison = energy_security_modeling_results.groupby(["country", "subsection"]).mean().loc[:,0:9]
```

```
In [82]: countries = energy_security_modeling_results.country.unique()
         sections = ["Policies and Measures", "National Objectives and Targets"]
```

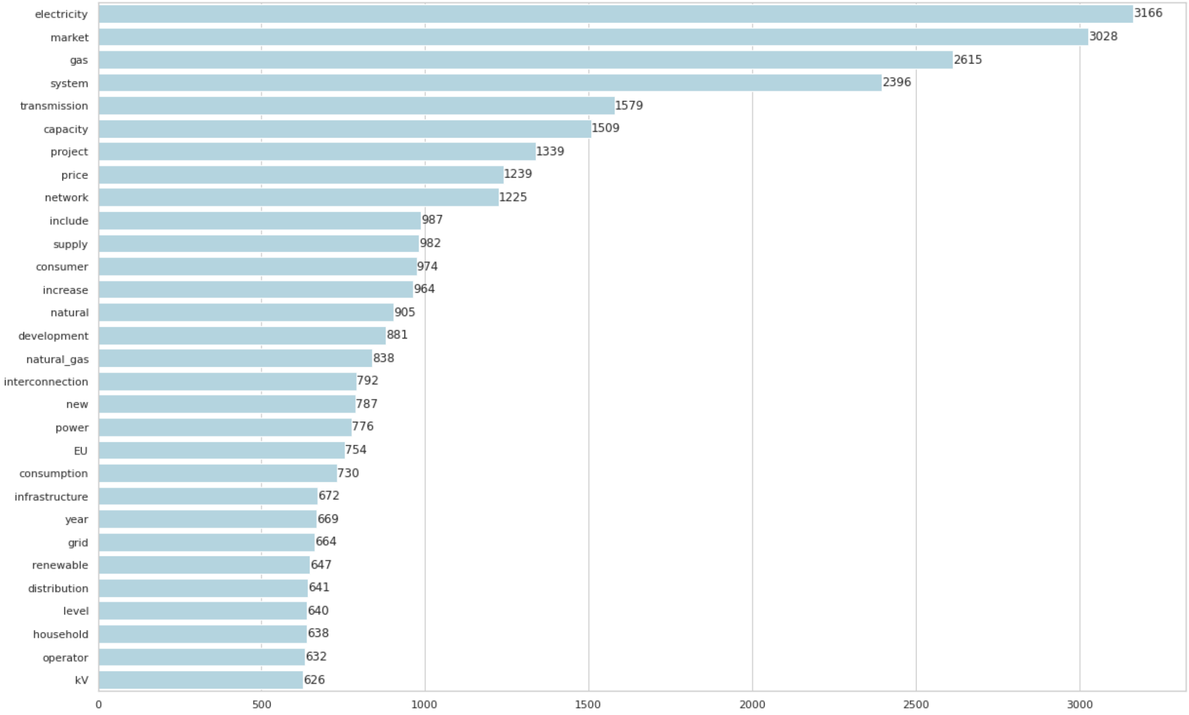
In [83]:
```
energy_security_change = {"country": [], "similarity": []}
for country in countries:
  pm = energy_security_modeling_results.loc[(energy_security_modeling_results["country"] == country) &
                                        (energy_security_modeling_results["subsection"] == sections[0])].loc[:,0:9]
  noat = energy_security_modeling_results.loc[(energy_security_modeling_results["country"] == country) &
                                        (energy_security_modeling_results["subsection"] == sections[1])].loc[:,0:9]
  if pm.shape[0]==1:
    energy_security_change["country"].append(country)
    energy_security_change["similarity"].append(1-sp.distance.cosine(pm, noat))
pd.DataFrame(energy_security_change)
```

Out[83]:

	country	similarity
0	Austria	0.949548
1	Belgium	1.000000
2	Bulgaria	0.942201
3	Czechia	0.929798
4	Cyprus	0.994925
5	Germany	0.648494
6	Denmark	0.925747
7	Estonia	1.000000
8	Croatia	0.996063
9	Finland	0.964796
10	France	0.414860
11	Malta	0.821248
12	Luxembourg	0.602821
13	Lithuania	0.999024
14	Latvia	0.807966
15	Italy	0.000000
16	Ireland	0.000000
17	Hungary	1.000000
18	Greece	0.999623
19	Spain	0.874892
20	Netherlands	0.870417
21	Poland	0.903131
22	Portugal	0.000000
23	Romania	0.914348
24	Sweden	0.761323
25	Slovenia	0.000000
26	Slovakia	0.981226

Dimension: Internal market

In [84]:
```
internal_market_docs = necp_processed[(necp_processed['energy_union_dimension'] == "Internal market")]["necp_lemmas"]
internal_market_counter = Counter(internal_market_docs.sum()).most_common(30)
plot_counter(internal_market_counter)
plt.show()
```



In [85]:
```
internal_market_docs = internal_market_docs.apply(lambda doc: [lemma for lemma in doc if not (lemma in ['electricity', 'market', 'gas', 'system'])])
```

```
In [86]: internal_market_dictionary = Dictionary(internal_market_docs)
         internal_market_dictionary.filter_extremes(no_below=2, no_above=1.0)
         internal_market_encoded_docs = internal_market_docs.apply(internal_market_dictionary.doc2bow)
```

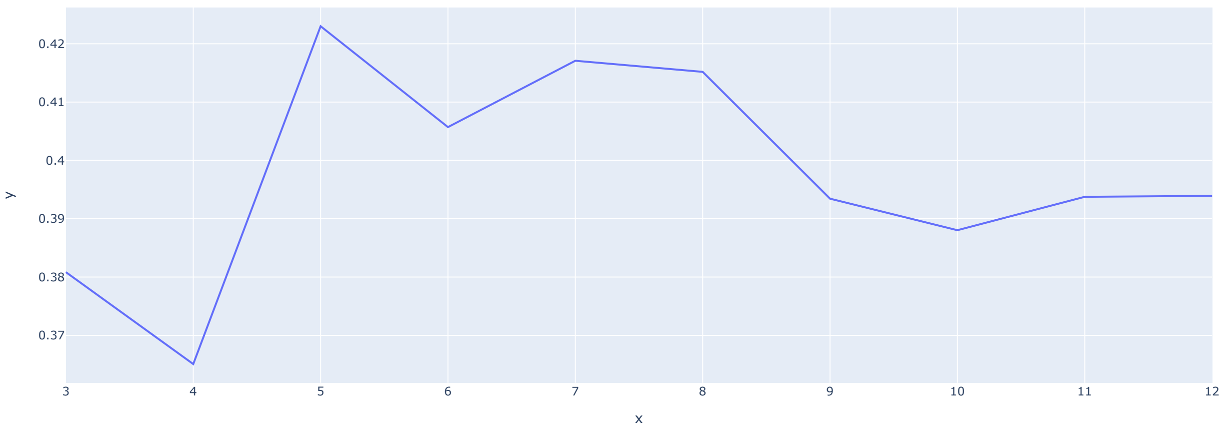
```
In [87]: internal_market_models = []
         for topics_number in tqdm(range(3, 13)):
             lda = LdaMulticore(internal_market_encoded_docs, num_topics=topics_number, passes=8, iterations=100, random_state=123)
             internal_market_models.append(lda)
```

```
  0%|          | 0/10 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning:

Calling np.sum(generator) is deprecated, and in the future will give a different result. Use np.sum(np.fromiter(generator)) or the python sum builtin instead.

100%|██████████| 10/10 [01:17<00:00,  7.76s/it]
```

```
In [88]: internal_market_cvs = []
         for model in tqdm(internal_market_models):
             cm = CoherenceModel(model,texts=internal_market_docs, dictionary=internal_market_dictionary)
             c_v = cm.get_coherence()
             internal_market_cvs.append(c_v)
```

```
100%|██████████| 10/10 [00:40<00:00,  4.08s/it]
```

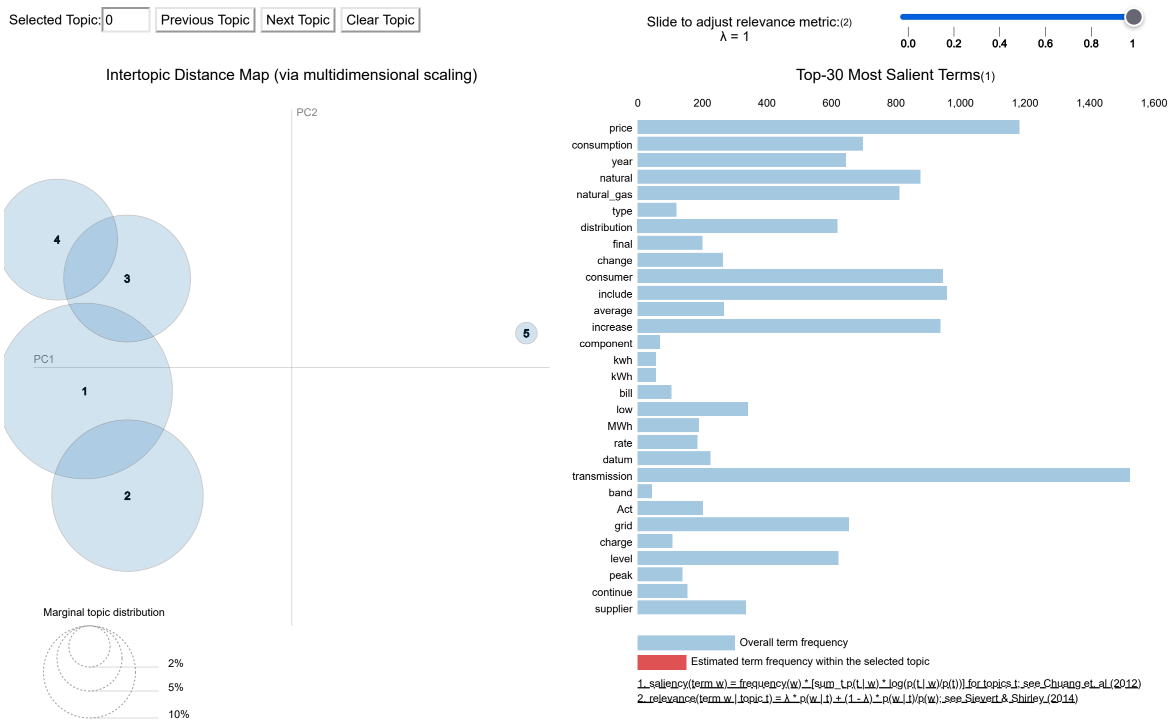
```
In [89]: px.line(x=range(3, 13), y=internal_market_cvs)
```



```
In [90]: vis = pyLDAvis.gensim_models.prepare(internal_market_models[2], internal_market_encoded_docs, dictionary=internal_market_dictionary)
         vis
```

```
/usr/local/lib/python3.7/dist-packages/pyLDAvis/_prepare.py:247: FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
```

Out[90]:



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

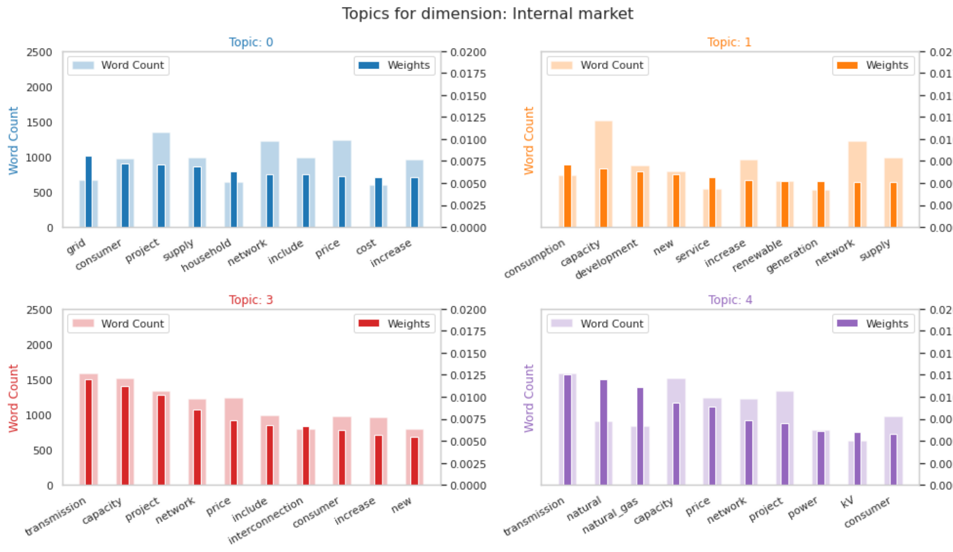
In [91]:
```
for idx, topic in internal_market_models[2].show_topics(formatted=False, num_words=15):
    print('Topic: {} \nWords: {}'.format(idx, [internal_market_dictionary[int(w[0])] for w in topic]))
```

```
Topic: 0
Words: ['grid', 'consumer', 'project', 'supply', 'household', 'network', 'include', 'price', 'cost', 'increase', 'Act', 'capacity', 'support', 'renewable', 'flexibility']
Topic: 1
Words: ['consumption', 'capacity', 'development', 'new', 'service', 'increase', 'renewable', 'generation', 'network', 'supply', 'storage', 'grid', 'project', 'interconnec
n', 'price']
Topic: 2
Words: ['kwh', 'band', 'price', 'kWh', 'consumption_band', 'consumption', 'Eurostat', 'component', 'DC', 'type', 'client', 'year', 'data', 'Kingdom', 'weighted']
Topic: 3
Words: ['transmission', 'capacity', 'project', 'network', 'price', 'include', 'interconnection', 'consumer', 'increase', 'new', 'supply', 'EU', 'line', 'development', 'kV
Topic: 4
Words: ['transmission', 'natural', 'natural_gas', 'capacity', 'price', 'network', 'project', 'power', 'kV', 'consumer', 'supply', 'increase', 'development', 'year', 'incl
']
```

- Topic 0: Act, grid, grid_expansion, Network Agency
- Topic 1: development, new, renewable
- Topic 2: band, consumption_band -- do wyrzucenia (0% tokenów)
- Topic 3: transmission, network, interconnection
- Topic 4: neutral, neutral gas, emission

In [92]:
```
from matplotlib import colors
topics = internal_market_models[2].show_topics(formatted=False)
counter = Counter(internal_market_docs.sum())

out = []
for i, topic in topics:
    for word, weight in topic:
        word = internal_market_dictionary[int(word)]
        out.append([word, i , weight, counter[word]])

df = pd.DataFrame(out, columns=['word', 'topic_id', 'importance', 'word_count'])

fig, axes = plt.subplots(2, 2, figsize=(14,8), sharey=True)
cols = [color for name, color in colors.TABLEAU_COLORS.items()]
for i, ax in enumerate(axes.flatten()):
    if i>=2:
      i+=1
    ax.bar(x='word', height="word_count", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.5, alpha=0.3, label='Word Count')
    ax_twin = ax.twinx()
    ax_twin.bar(x='word', height="importance", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.2, label='Weights')
    ax.set_ylabel('Word Count', color=cols[i])
    ax_twin.set_ylim(0, 0.02);
    ax.set_ylim(0, 2500)
    ax.set_title('Topic: ' + str(i), color=cols[i], fontsize=12)
    ax.tick_params(axis='y', left=False)
    ax.set_xticklabels(df.loc[df.topic_id==i, 'word'], rotation=30, horizontalalignment= 'right')
    ax.legend(loc='upper left'); ax_twin.legend(loc='upper right')
    ax.grid(False)
    ax_twin.grid(False)
fig.suptitle('Topics for dimension: Internal market', fontsize=16)
fig.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:26: UserWarning:

FixedFormatter should only be used together with FixedLocator
```



In [93]:
```
internal_market_corpus_model = internal_market_models[2][internal_market_encoded_docs]
```

In [94]:
```
internal_market_metainfo = necp_processed[(necp_processed['energy_union_dimension'] == "Internal market")]
res_len = len(internal_market_metainfo)
res = np.zeros((res_len, 5))
```

In [95]:
```
for i, doc in enumerate(internal_market_corpus_model):
  for topic in doc:
    res[i][topic[0]] = np.round(topic[1], 4)
```

In [96]:
```
internal_market_modeling_results = pd.concat([internal_market_metainfo.reset_index(drop=True), pd.DataFrame(res)], axis=1)
internal_market_topic_probs = internal_market_modeling_results.groupby("country").mean().loc[:,[0, 1, 2, 4]]
```

```
In [97]: internal_market_topic_probs
```

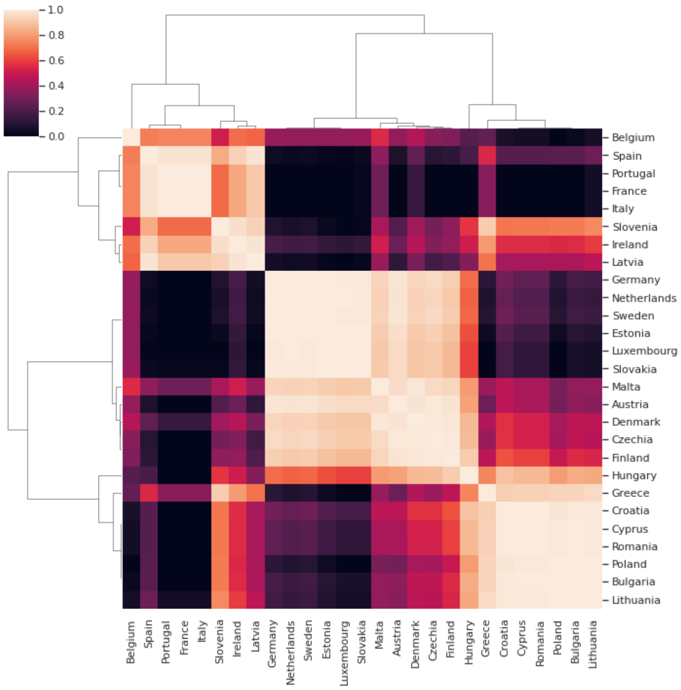
Out[97]:

country	0	1	2	4
Austria	0.741467	0.000000	0.019733	0.228867
Belgium	0.221633	0.444867	0.332933	0.000000
Bulgaria	0.054033	0.000000	0.000000	0.789967
Croatia	0.118733	0.000000	0.000000	0.648533
Cyprus	0.096867	0.000000	0.000000	0.745433
Czechia	0.028867	0.000000	0.000000	0.012967
Denmark	0.271267	0.044500	0.000000	0.124133
Estonia	0.355767	0.000000	0.000000	0.012967
Finland	0.186633	0.000000	0.000000	0.105367
France	0.000000	0.974300	0.000000	0.000000
Germany	0.768033	0.000000	0.012200	0.092767
Greece	0.000000	0.229150	0.000000	0.629900
Hungary	0.365667	0.000000	0.000000	0.483233
Ireland	0.064900	0.454367	0.034767	0.298467
Italy	0.000000	0.334167	0.000000	0.000000
Latvia	0.000000	0.445300	0.000000	0.206067
Lithuania	0.033467	0.032067	0.000000	0.643933
Luxembourg	0.645600	0.000000	0.000000	0.000000
Malta	0.139133	0.043433	0.000000	0.047067
Netherlands	0.919400	0.000000	0.000000	0.080100
Poland	0.000000	0.000000	0.000000	0.333267
Portugal	0.000000	0.007900	0.000000	0.000000
Romania	0.054833	0.000000	0.000000	0.424733
Slovakia	0.273000	0.004867	0.000000	0.000000
Slovenia	0.000000	0.474533	0.000000	0.498967
Spain	0.003367	0.305267	0.000000	0.071167
Sweden	0.339733	0.000000	0.000000	0.035500

```
In [98]: linkage = hc.linkage(internal_market_topic_probs, method='average', metric='cosine')
         internal_market_similarities = sp.distance.squareform(sp.distance.pdist(internal_market_topic_probs.values, metric='cosine'))
```

```
In [99]: plt.figure(figsize=(12, 8))
         sns.clustermap(1-internal_market_similarities,
                    xticklabels=internal_market_topic_probs.index,
                    yticklabels=internal_market_topic_probs.index,
                     row_linkage=linkage, col_linkage=linkage)
         plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```



```
In [100]: internal_market_comparison = internal_market_modeling_results.groupby(["country", "subsection"]).mean().loc[:,0:4]
```

```
In [101]: countries = internal_market_modeling_results.country.unique()
          sections = ["Policies and Measures", "National Objectives and Targets"]
```

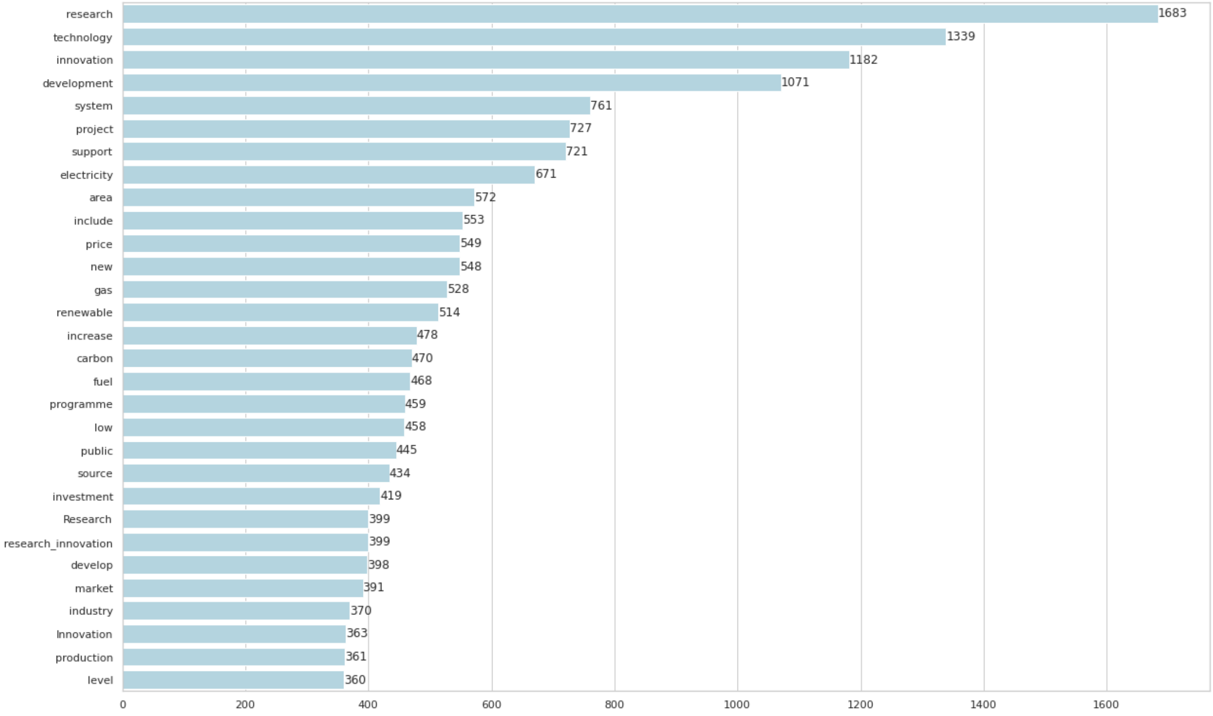
In [102]:
```
internal_market_change = {"country": [], "similarity": []}
for country in countries:
  pm = internal_market_modeling_results.loc[(internal_market_modeling_results["country"] == country) &
                                           (internal_market_modeling_results["subsection"] == sections[0])].loc[:,0:4]
  noat = internal_market_modeling_results.loc[(internal_market_modeling_results["country"] == country) &
                                           (internal_market_modeling_results["subsection"] == sections[1])].loc[:,0:4]
  if pm.shape[0]==1:
    internal_market_change["country"].append(country)
    internal_market_change["similarity"].append(1-sp.distance.cosine(pm, noat))
pd.DataFrame(internal_market_change)
```

Out[102]:

	country	similarity
0	Austria	0.910670
1	Belgium	0.536360
2	Bulgaria	0.738063
3	Czechia	0.995534
4	Cyprus	0.702632
5	Germany	1.000000
6	Denmark	0.362179
7	Estonia	0.436535
8	Croatia	0.399743
9	Finland	0.864747
10	France	0.996828
11	Malta	0.703933
12	Luxembourg	0.997848
13	Lithuania	0.174850
14	Latvia	0.740835
15	Italy	0.682852
16	Ireland	0.873506
17	Hungary	0.692078
18	Greece	0.834239
19	Spain	0.999485
20	Netherlands	1.000000
21	Poland	1.000000
22	Portugal	0.999705
23	Romania	0.865309
24	Sweden	0.020312
25	Slovenia	0.645319
26	Slovakia	0.198617

Dimension: R&I and Competitiveness

In [103]:
```
research_docs = necp_processed[(necp_processed['energy_union_dimension'] == "R&I and Competitiveness")]["necp_lemmas"]
research_counter = Counter(research_docs.sum()).most_common(30)
plot_counter(research_counter)
plt.show()
```



In [104]:
```
research_docs = research_docs.apply(lambda doc: [lemma for lemma in doc if not (lemma in ['research'])])
```

```
In [105]: research_dictionary = Dictionary(research_docs)
          research_dictionary.filter_extremes(no_below=2, no_above=1.0)
          research_encoded_docs = research_docs.apply(research_dictionary.doc2bow)
```

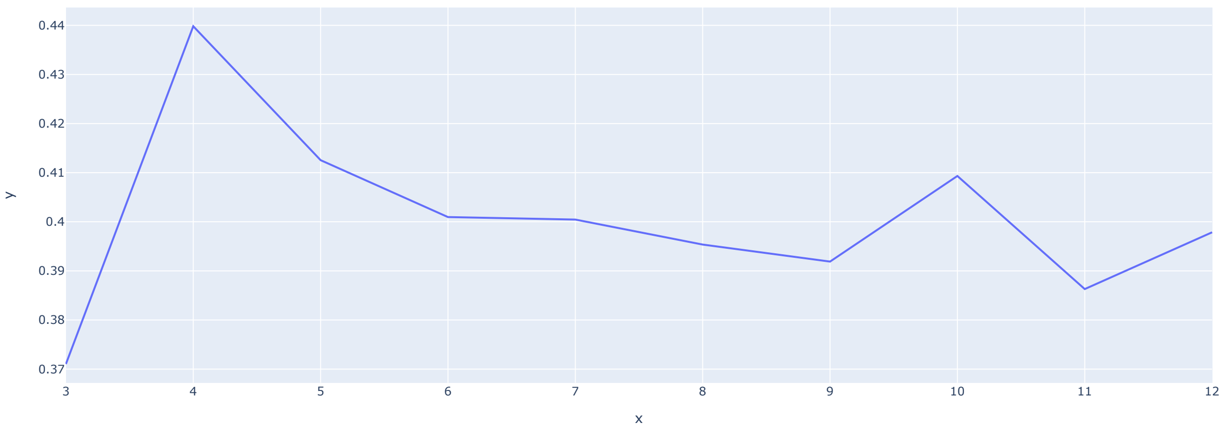
```
In [106]: research_models = []
          for topics_number in tqdm(range(3, 13)):
              lda = LdaMulticore(research_encoded_docs, num_topics=topics_number, passes=10, iterations=80, random_state=42)
              research_models.append(lda)
```

```
  0%|          | 0/10 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning:

Calling np.sum(generator) is deprecated, and in the future will give a different result. Use np.sum(np.fromiter(generator)) or the python sum builtin instead.

100%|██████████| 10/10 [01:07<00:00,  6.74s/it]
```

```
In [107]: research_cvs = []
          for model in tqdm(research_models):
              cm = CoherenceModel(model,texts=research_docs, dictionary=research_dictionary)
              c_v = cm.get_coherence()
              research_cvs.append(c_v)
```

```
100%|██████████| 10/10 [00:31<00:00,  3.15s/it]
```

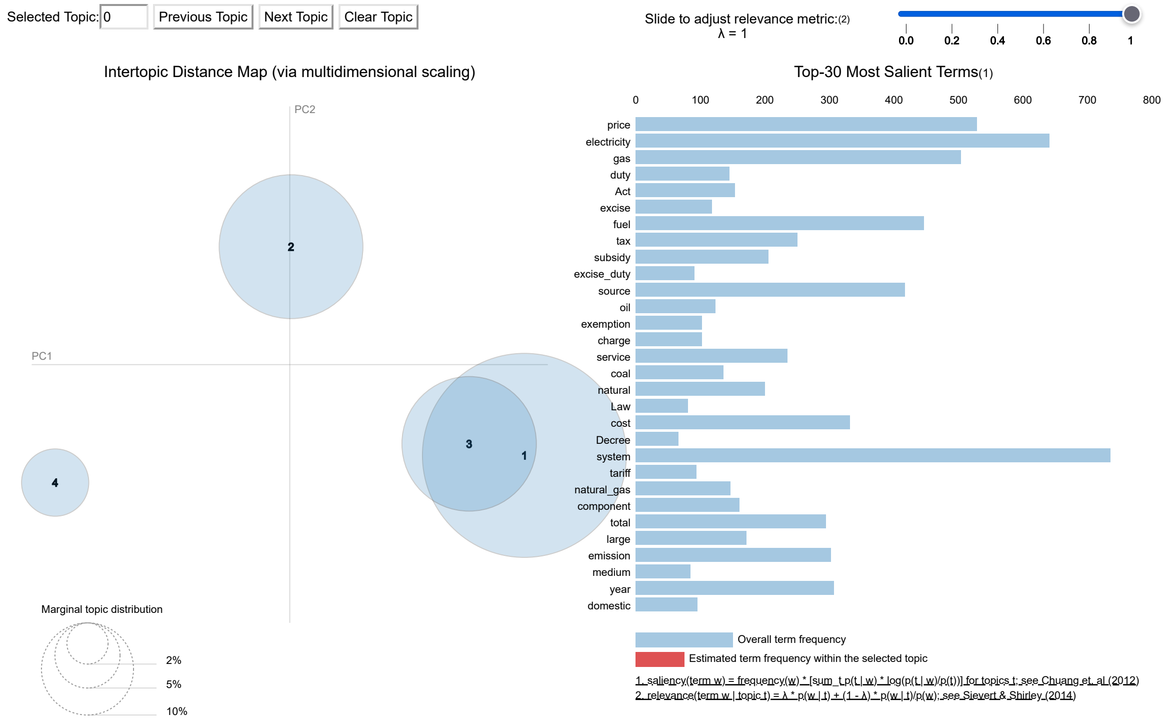
```
In [108]: px.line(x=range(3, 13), y=research_cvs)
```



```
In [109]: vis = pyLDAvis.gensim_models.prepare(research_models[1], research_encoded_docs, dictionary=research_dictionary)
          vis
```

```
/usr/local/lib/python3.7/dist-packages/pyLDAvis/_prepare.py:247: FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
```

Out[109]:



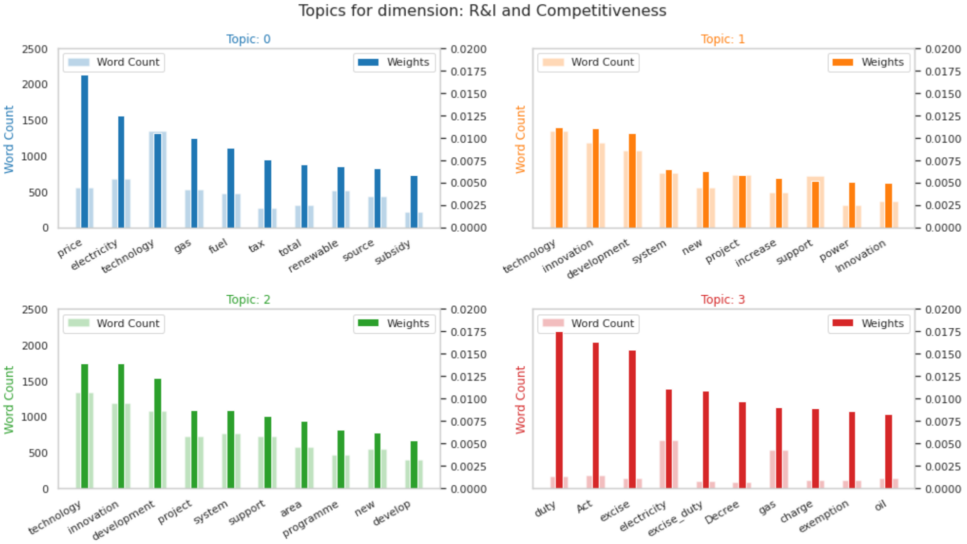
In [110]:
```
for idx, topic in research_models[1].show_topics(formatted=False, num_words=15):
    print('Topic: {} \nWords: {}'.format(idx, [research_dictionary[int(w[0])] for w in topic]))
```

```
Topic: 0
Words: ['price', 'electricity', 'technology', 'gas', 'fuel', 'tax', 'total', 'renewable', 'source', 'subsidy', 'low', 'investment', 'EUR', 'support', 'cost']
Topic: 1
Words: ['technology', 'innovation', 'development', 'system', 'new', 'project', 'increase', 'support', 'power', 'Innovation', 'include', 'Research', 'area', 'renewable', '
ctricity']
Topic: 2
Words: ['technology', 'innovation', 'development', 'project', 'system', 'support', 'area', 'programme', 'new', 'develop', 'include', 'research_innovation', 'competitivene
', 'carbon', 'public']
Topic: 3
Words: ['duty', 'Act', 'excise', 'electricity', 'excise_duty', 'Decree', 'gas', 'charge', 'exemption', 'oil', 'Law', 'fuel', 'coal', 'source', 'system']
```

- Topic 0: price, electricity, gas, fuel, source, tax, subsidy, expenditure, household
- Topic 1: innovation, technology, power, nuclear power, water, renewable, emission
- Topic 2: research innovation, technology, development, programme, cooperation, project, competitiveness
- Topic 3: duty, Act, excise, law, charge

In [111]:
```
from matplotlib import colors
topics = research_models[1].show_topics(formatted=False)
counter = Counter(research_docs.sum())

out = []
for i, topic in topics:
    for word, weight in topic:
        word = research_dictionary[int(word)]
        out.append([word, i , weight, counter[word]])

df = pd.DataFrame(out, columns=['word', 'topic_id', 'importance', 'word_count'])

fig, axes = plt.subplots(2, 2, figsize=(14,8), sharey=True)
cols = [color for name, color in colors.TABLEAU_COLORS.items()]
for i, ax in enumerate(axes.flatten()):
    ax.bar(x='word', height="word_count", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.5, alpha=0.3, label='Word Count')
    ax_twin = ax.twinx()
    ax_twin.bar(x='word', height="importance", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.2, label='Weights')
    ax.set_ylabel('Word Count', color=cols[i])
    ax_twin.set_ylim(0, 0.02); ax.set_ylim(0, 2500)
    ax.set_title('Topic: ' + str(i), color=cols[i], fontsize=12)
    ax.tick_params(axis='y', left=False)
    ax.set_xticklabels(df.loc[df.topic_id==i, 'word'], rotation=30, horizontalalignment= 'right')
    ax.legend(loc='upper left'); ax_twin.legend(loc='upper right')
    ax.grid(False)
    ax_twin.grid(False)
fig.suptitle('Topics for dimension: R&I and Competitiveness', fontsize=16)
fig.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:23: UserWarning:

FixedFormatter should only be used together with FixedLocator
```



In [112]:
```
research_corpus_model = research_models[1][research_encoded_docs]
```

In [113]:
```
research_metainfo = necp_processed[(necp_processed['energy_union_dimension'] == "R&I and Competitiveness")]
res_len = len(research_metainfo)
res = np.zeros((res_len, 4))
```

In [114]:
```
for i, doc in enumerate(research_corpus_model):
  for topic in doc:
    res[i][topic[0]] = np.round(topic[1], 4)
```

In [115]:
```
research_modeling_results = pd.concat([research_metainfo.reset_index(drop=True), pd.DataFrame(res)], axis=1)
research_topic_probs = research_modeling_results.groupby("country").mean().loc[:,[0, 1, 2, 3]]
```

In [116]:
```
research_topic_probs
```

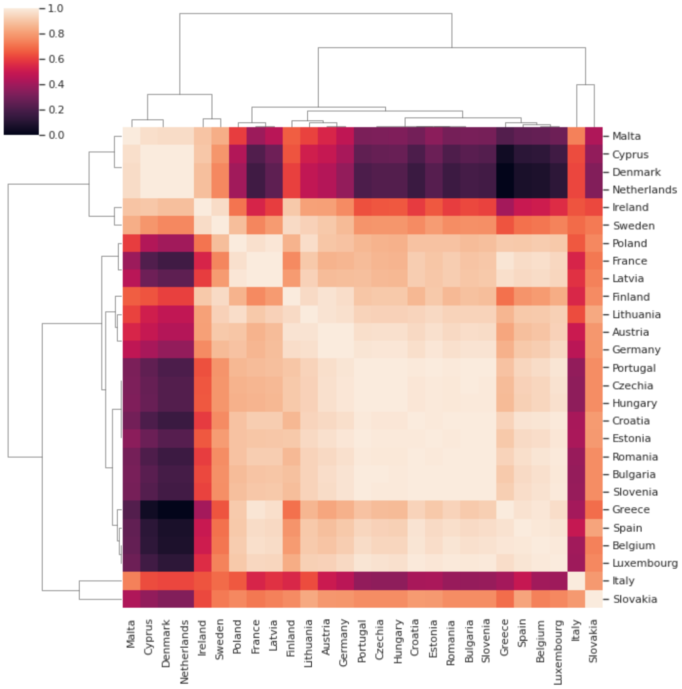
Out[116]:

country	0	1	2	3
Austria	0.328567	0.149133	0.522000	0.000000
Belgium	0.146433	0.000000	0.852733	0.000000
Bulgaria	0.308200	0.000000	0.690300	0.000000
Croatia	0.257900	0.000000	0.700067	0.041267
Cyprus	0.332967	0.632767	0.032267	0.000000
Czechia	0.355300	0.000000	0.644267	0.000000
Denmark	0.333000	0.665967	0.000000	0.000000
Estonia	0.297100	0.031233	0.645633	0.022300
Finland	0.362633	0.219733	0.415800	0.000000
France	0.000000	0.161933	0.836700	0.000000
Germany	0.333133	0.106733	0.559600	0.000000
Greece	0.000000	0.000000	0.999150	0.000000
Hungary	0.358300	0.000000	0.639867	0.000000
Ireland	0.352667	0.405667	0.241233	0.000000
Italy	0.000000	0.406100	0.240533	0.353133
Latvia	0.033467	0.198667	0.766400	0.000000
Lithuania	0.225800	0.208533	0.522967	0.042267
Luxembourg	0.217233	0.000000	0.781267	0.000000
Malta	0.189400	0.658067	0.151533	0.000000
Netherlands	0.333233	0.666367	0.000000	0.000000
Poland	0.076967	0.276700	0.639967	0.005833
Portugal	0.333167	0.000000	0.665733	0.000000
Romania	0.273700	0.000000	0.725800	0.000000
Slovakia	0.176700	0.112067	0.377233	0.333233
Slovenia	0.294000	0.000000	0.705100	0.000000
Spain	0.122200	0.000000	0.760533	0.116933
Sweden	0.263900	0.359600	0.375767	0.000000

In [117]:
```
import scipy.spatial as sp
import scipy.cluster.hierarchy as hc
linkage = hc.linkage(research_topic_probs, method='average', metric='cosine')
research_similarities = sp.distance.squareform(sp.distance.pdist(research_topic_probs.values, metric='cosine'))
```

In [118]:
```
plt.figure(figsize=(12, 8))
sns.clustermap(1-research_similarities,
            xticklabels=research_topic_probs.index,
            yticklabels=research_topic_probs.index,
             row_linkage=linkage, col_linkage=linkage)
plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```



In [119]:
```
research_comparison = research_modeling_results.groupby(["country", "subsection"]).mean().loc[:,0:3]
```

In [120]:
```
countries = research_modeling_results.country.unique()
sections = ["Policies and Measures", "National Objectives and Targets"]
```

```
In [121]: research_change = {"country": [], "similarity": []}
          for country in countries:
            pm = research_modeling_results.loc[(research_modeling_results["country"] == country) &
                                               (research_modeling_results["subsection"] == sections[0])].loc[:,0:3]
            noat = research_modeling_results.loc[(research_modeling_results["country"] == country) &
                                               (research_modeling_results["subsection"] == sections[1])].loc[:,0:3]
            if pm.shape[0]==1:
              research_change["country"].append(country)
              research_change["similarity"].append(1-sp.distance.cosine(pm, noat))
          pd.DataFrame(research_change)
```

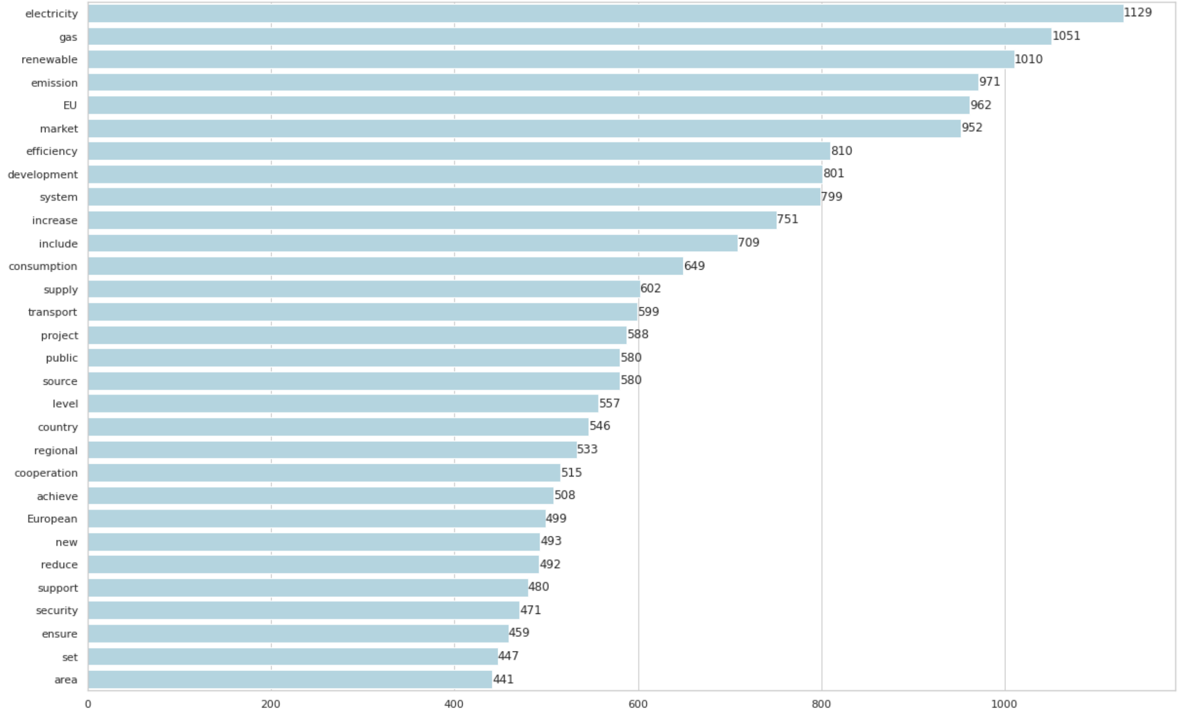
Out[121]:

	country	similarity
0	Austria	0.759601
1	Belgium	1.000000
2	Bulgaria	1.000000
3	Czechia	0.999862
4	Cyprus	0.994300
5	Germany	0.904498
6	Denmark	1.000000
7	Estonia	0.994688
8	Croatia	1.000000
9	Finland	0.937662
10	France	0.726629
11	Malta	0.996773
12	Luxembourg	1.000000
13	Lithuania	0.915952
14	Latvia	0.983657
15	Italy	0.927599
16	Ireland	0.757675
17	Hungary	0.996632
18	Greece	1.000000
19	Spain	1.000000
20	Netherlands	1.000000
21	Poland	0.997626
22	Portugal	1.000000
23	Romania	0.993925
24	Sweden	0.227024
25	Slovenia	1.000000
26	Slovakia	0.493654

```
In [ ]:
```

subsection: Overview and Process for Establishing the Plan

```
In [122]: overview_docs = necp_processed[(necp_processed['subsection'] == "Overview and Process for Establishing the Plan")]["necp_lemmas"]
          overview_counter = Counter(overview_docs.sum()).most_common(30)
          plot_counter(overview_counter)
          plt.show()
```



```
In [123]: overview_docs = overview_docs.apply(lambda doc: [lemma for lemma in doc if not (lemma in ['emission', 'renewable'])])
```

In [124]:
```
overview_dictionary = Dictionary(overview_docs)
overview_dictionary.filter_extremes(no_below=2, no_above=1.0)
overview_encoded_docs = overview_docs.apply(overview_dictionary.doc2bow)
```

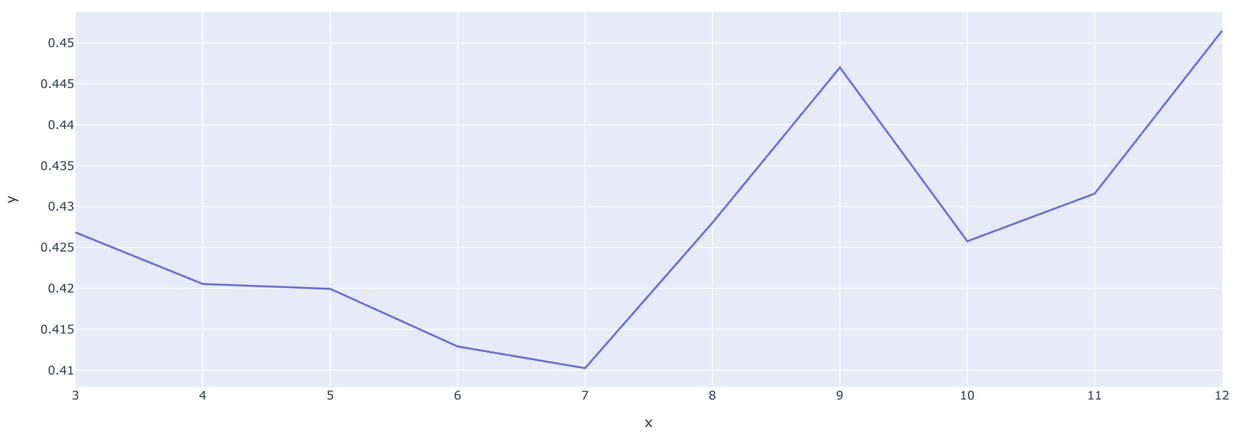
In [125]:
```
overview_models = []
for topics_number in tqdm(range(3, 13)):
    lda = LdaMulticore(overview_encoded_docs, num_topics=topics_number, passes=8, iterations=100, random_state=123)
    overview_models.append(lda)
```

```
  0%|          | 0/10 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning:

Calling np.sum(generator) is deprecated, and in the future will give a different result. Use np.sum(np.fromiter(generator)) or the python sum builtin instead.

100%|██████████| 10/10 [00:51<00:00,  5.14s/it]
```

In [126]:
```
overview_cvs = []
for model in tqdm(overview_models):
    cm = CoherenceModel(model,texts=overview_docs, dictionary=overview_dictionary)
    c_v = cm.get_coherence()
    overview_cvs.append(c_v)
```

```
100%|██████████| 10/10 [00:36<00:00,  3.66s/it]
```

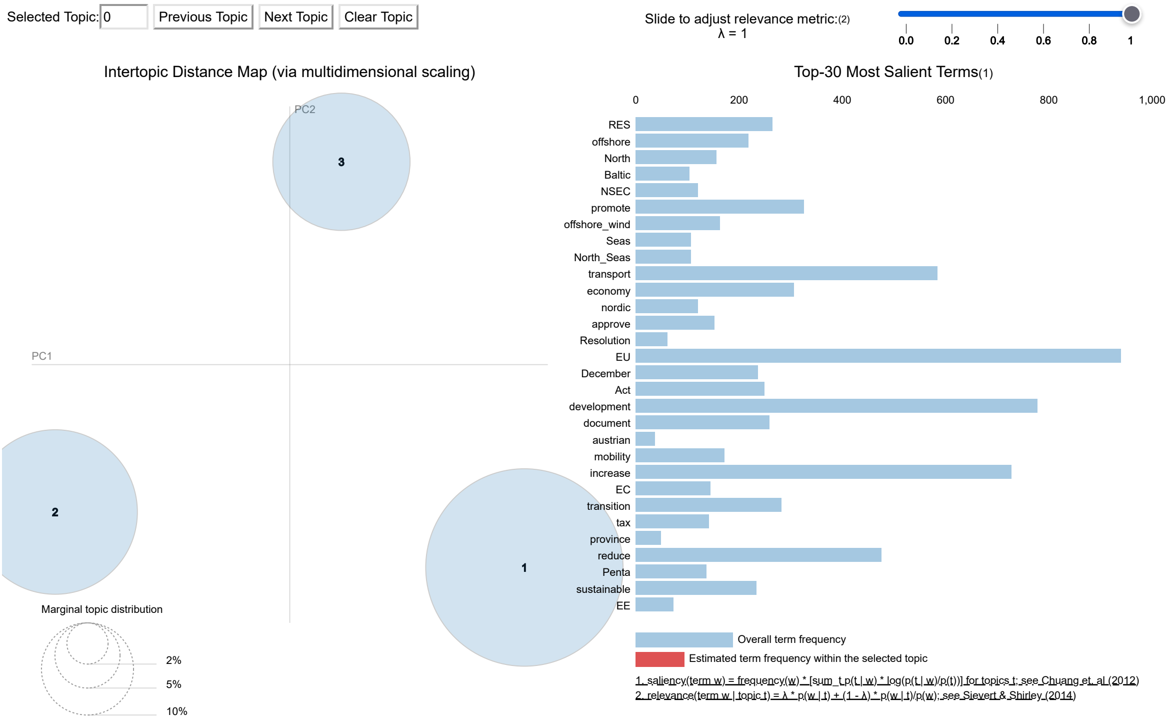
In [127]:
```
px.line(x=range(3, 13), y=overview_cvs)
```



In [128]:
```
vis = pyLDAvis.gensim_models.prepare(overview_models[0], overview_encoded_docs, dictionary=overview_dictionary)
vis
```

```
/usr/local/lib/python3.7/dist-packages/pyLDAvis/_prepare.py:247: FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
```

Out[128]:



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

In [129]:
```
for idx, topic in overview_models[0].show_topics(formatted=False, num_words=15):
    print('Topic: {} \nWords: {}'.format(idx, [overview_dictionary[int(w[0])] for w in topic]))
```

```
Topic: 0
Words: ['electricity', 'gas', 'market', 'EU', 'system', 'supply', 'efficiency', 'project', 'country', 'include', 'development', 'increase', 'consumption', 'regional', 'wo
']
Topic: 1
Words: ['gas', 'electricity', 'efficiency', 'market', 'increase', 'development', 'system', 'EU', 'reduce', 'transport', 'promote', 'consumption', 'supply', 'level', 'sour
']
Topic: 2
Words: ['EU', 'electricity', 'gas', 'development', 'include', 'transport', 'market', 'efficiency', 'increase', 'system', 'source', 'consumption', 'RES', 'public', 'Minist
']
```

- Topic 0: electricity, emission, renewable, system, country, regional
- Topic 1: project, wind, cooperation, nordic
- Topic 2: gas, market, efficiency, source, public, transport

In [130]:
```
overview_corpus_model = overview_models[0][overview_encoded_docs]
```

In [131]:
```
overview_metainfo = necp_processed[(necp_processed['subsection'] == "Overview and Process for Establishing the Plan")]
res_len = len(overview_metainfo)
res = np.zeros((res_len, 3))
```

In [132]:
```
for i, doc in enumerate(overview_corpus_model):
  for topic in doc:
    res[i][topic[0]] = np.round(topic[1], 4)
```

In [133]:
```
overview_modeling_results = pd.concat([overview_metainfo.reset_index(drop=True), pd.DataFrame(res)], axis=1)
overview_topic_probs = overview_modeling_results.groupby("country").mean().loc[:,[0, 1, 2]]
```

In [134]:
```
overview_topic_probs
```

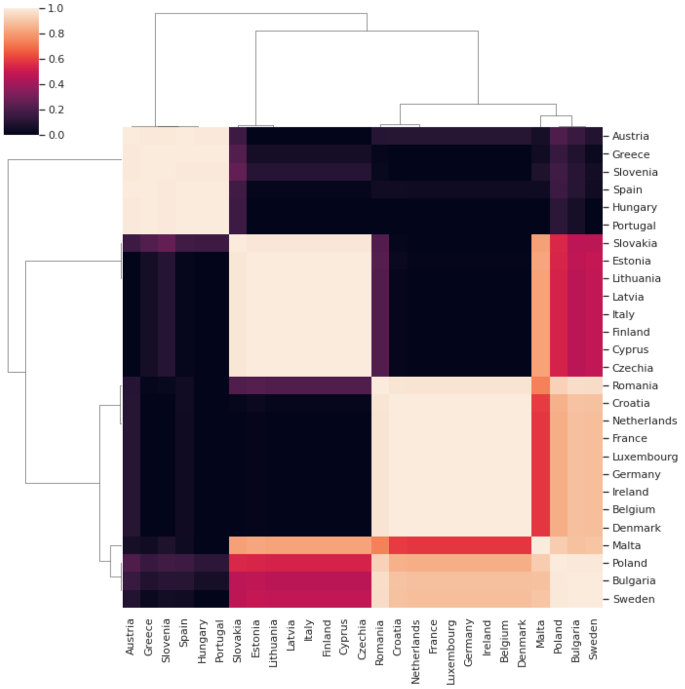
Out[134]:

	0	1	2
country			
Austria	0.0931	0.9069	0.0000
Belgium	0.9989	0.0000	0.0000
Bulgaria	0.6294	0.0432	0.3273
Croatia	0.9858	0.0000	0.0142
Cyprus	0.0000	0.0000	0.9985
Czechia	0.0000	0.0000	0.9998
Denmark	0.9997	0.0000	0.0000
Estonia	0.0106	0.0000	0.9893
Finland	0.0000	0.0000	0.9954
France	0.9999	0.0000	0.0000
Germany	0.9999	0.0000	0.0000
Greece	0.0000	0.9457	0.0542
Hungary	0.0000	0.9974	0.0000
Ireland	0.9998	0.0000	0.0000
Italy	0.0000	0.0000	0.9973
Latvia	0.0000	0.0000	0.9997
Lithuania	0.0000	0.0000	0.9997
Luxembourg	0.9998	0.0000	0.0000
Malta	0.4133	0.0000	0.5831
Netherlands	0.9999	0.0000	0.0000
Poland	0.5652	0.0795	0.3554
Portugal	0.0000	0.9999	0.0000
Romania	0.8228	0.0000	0.1771
Slovakia	0.0000	0.1374	0.8625
Slovenia	0.0000	0.9053	0.0914
Spain	0.0405	0.9460	0.0136
Sweden	0.6504	0.0000	0.3493

In [135]:
```
import scipy.spatial as sp
import scipy.cluster.hierarchy as hc
linkage = hc.linkage(overview_topic_probs, method='average', metric='cosine')
overview_similarities = sp.distance.squareform(sp.distance.pdist(overview_topic_probs.values, metric='cosine'))
```

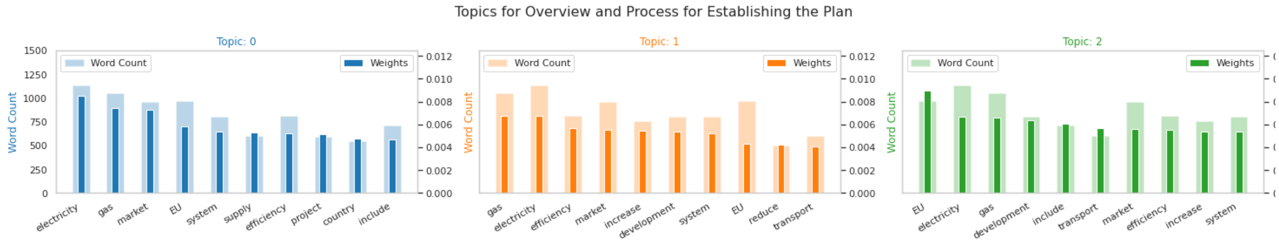
In [136]:
```
plt.figure(figsize=(12, 8))
sns.clustermap(1-overview_similarities,
            xticklabels=overview_topic_probs.index,
            yticklabels=overview_topic_probs.index,
             row_linkage=linkage, col_linkage=linkage)
plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```



In [137]:
```
from matplotlib import colors
topics = overview_models[0].show_topics(formatted=False)
counter = Counter(overview_docs.sum())

out = []
for i, topic in topics:
    for word, weight in topic:
        word = overview_dictionary[int(word)]
        out.append([word, i , weight, counter[word]])

df = pd.DataFrame(out, columns=['word', 'topic_id', 'importance', 'word_count'])

fig, axes = plt.subplots(1, 3, figsize=(21, 4), sharey=True)
cols = [color for name, color in colors.TABLEAU_COLORS.items()]

for i, ax in enumerate(axes.flatten()):
    ax.bar(x='word', height="word_count", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.5, alpha=0.3, label='Word Count')
    ax_twin = ax.twinx()
    ax_twin.bar(x='word', height="importance", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.2, label='Weights')
    ax.set_ylabel('Word Count', color=cols[i])
    ax_twin.set_ylim(0, 0.0125); ax.set_ylim(0, 1500)
    ax.set_title('Topic: ' + str(i), color=cols[i], fontsize=12)
    ax.tick_params(axis='y', left=False)
    ax.set_xticklabels(df.loc[df.topic_id==i, 'word'], rotation=30, horizontalalignment= 'right')
    ax.legend(loc='upper left'); ax_twin.legend(loc='upper right')
    ax.grid(False)
    ax_twin.grid(False)
fig.suptitle('Topics for Overview and Process for Establishing the Plan', fontsize=16)
fig.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:24: UserWarning:

FixedFormatter should only be used together with FixedLocator
```



Overview and Process for Establishing the Plan with the removal of most common lemmas

In [138]:
```
overview_docs = overview_docs.apply(lambda doc: [lemma for lemma in doc if not (lemma in ['electricity', 'gas', 'renewable', 'emission'])])
```

In [139]:
```
overview_dictionary = Dictionary(overview_docs)
overview_dictionary.filter_extremes(no_below=2, no_above=1.0)
overview_encoded_docs = overview_docs.apply(overview_dictionary.doc2bow)
```

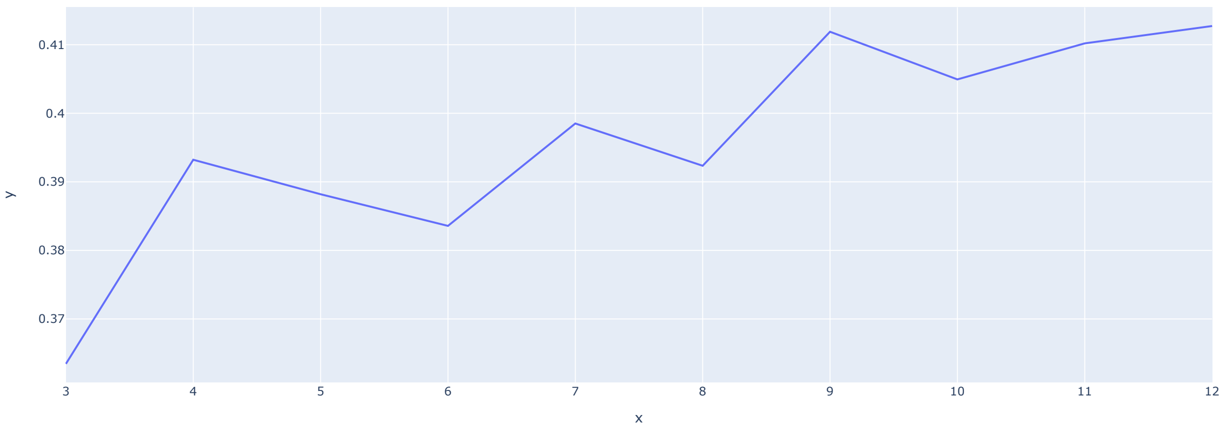
```
In [140]: overview_models = []
          for topics_number in tqdm(range(3, 13)):
              lda = LdaMulticore(overview_encoded_docs, num_topics=topics_number, passes=8, iterations=100, random_state=123)
              overview_models.append(lda)
```

```
  0%|          | 0/10 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning:

Calling np.sum(generator) is deprecated, and in the future will give a different result. Use np.sum(np.fromiter(generator)) or the python sum builtin instead.

100%|██████████| 10/10 [00:50<00:00,  5.01s/it]
```

```
In [141]: overview_cvs = []
          for model in tqdm(overview_models):
              cm = CoherenceModel(model,texts=overview_docs, dictionary=overview_dictionary)
              c_v = cm.get_coherence()
              overview_cvs.append(c_v)
```

```
100%|██████████| 10/10 [00:36<00:00,  3.68s/it]
```

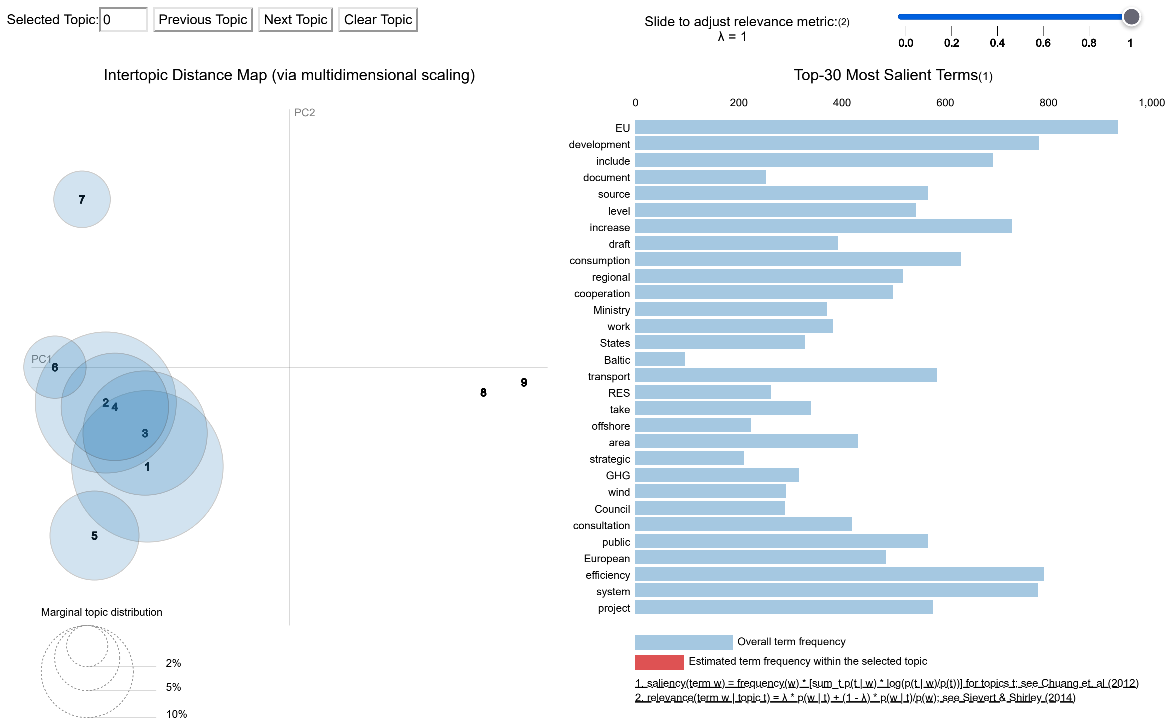
```
In [142]: px.line(x=range(3, 13), y=overview_cvs)
```



```
In [143]: vis = pyLDAvis.gensim_models.prepare(overview_models[6], overview_encoded_docs, dictionary=overview_dictionary)
          vis
```

```
/usr/local/lib/python3.7/dist-packages/pyLDAvis/_prepare.py:247: FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
```

Out[143]:



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

In [144]:
```
for idx, topic in overview_models[6].show_topics(formatted=False, num_words=15):
    print('Topic: {} \nWords: {}'.format(idx, [overview_dictionary[int(w[0])] for w in topic]))
```

```
Topic: 0
Words: ['work', 'EU', 'country', 'offshore', 'consultation', 'set', 'market', 'include', 'regional', 'develop', 'wind', 'support', 'supply', 'Commission', 'level']
Topic: 1
Words: ['market', 'cooperation', 'system', 'EU', 'efficiency', 'project', 'country', 'development', 'building', 'increase', 'regional', 'work', 'include', 'public', 'new'
Topic: 2
Words: ['EU', 'efficiency', 'transport', 'market', 'development', 'system', 'area', 'source', 'reduce', 'increase', 'achieve', 'project', 'include', 'country', 'public']
Topic: 3
Words: ['market', 'development', 'efficiency', 'system', 'EU', 'source', 'include', 'consumption', 'transport', 'project', 'increase', 'RES', 'public', 'draft', 'natural'
Topic: 4
Words: ['efficiency', 'increase', 'process', 'system', 'promote', 'market', 'development', 'reduce', 'consumption', 'economy', 'new', 'establish', 'achieve', 'EU', 'level
Topic: 5
Words: ['EU', 'Baltic', 'development', 'include', 'draft', 'consumption', 'States', 'regional', 'cooperation', 'Council', 'GHG', 'Ministry', 'transport', 'final', 'Europe
']
Topic: 6
Words: ['document', 'level', 'source', 'development', 'increase', 'strategic', 'EU', 'include', 'area', 'State', 'take', 'strategic_document', 'regional', 'information',
nsumption']
Topic: 7
Words: ['EU', 'system', 'include', 'market', 'public', 'project', 'increase', 'efficiency', 'development', 'transport', 'supply', 'support', 'security', 'source', 'achiev
Topic: 8
Words: ['EU', 'supply', 'increase', 'system', 'transport', 'efficiency', 'market', 'consumption', 'source', 'reduce', 'country', 'include', 'level', 'development', 'share
```

- Topic 0: Region, offshore, work, wind, seas
- Topic 1: cooperation, market, building, nordic, north, wind
- Topic 2: 0%
- Topic 3: INECP, RES, development, market, system, EE
- Topic 4: efficiency, increase, Resolution, approve
- Topic 5: Baltic, development, officials
- Topic 6: document, level, strategic, source
- Topic 7: 0%
- Topic 8: supply, increase, tax, province

In [145]:
```
overview_corpus_model = overview_models[6][overview_encoded_docs]
```

In [146]:
```
overview_metainfo = necp_processed[(necp_processed['subsection'] == "Overview and Process for Establishing the Plan")]
res_len = len(overview_metainfo)
res = np.zeros((res_len, 10))
```

In [147]:
```
for i, doc in enumerate(overview_corpus_model):
  for topic in doc:
    res[i][topic[0]] = np.round(topic[1], 4)
```

In [148]:
```
overview_modeling_results = pd.concat([overview_metainfo.reset_index(drop=True), pd.DataFrame(res)], axis=1)
overview_topic_probs = overview_modeling_results.groupby("country").mean().loc[:,[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]]
```

In [149]:
```
overview_topic_probs
```

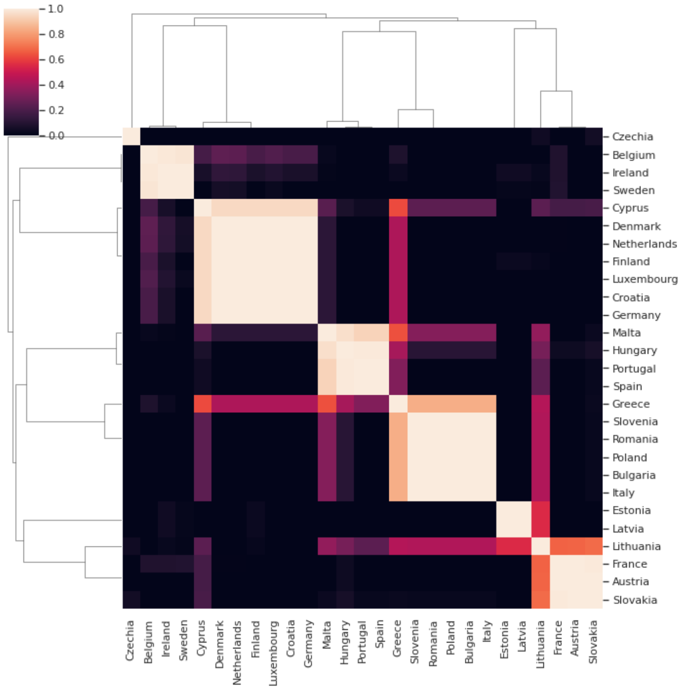
Out[149]:

	0	1	2	3	4	5	6	7	8	9
country										
Austria	0.0000	0.0000	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.9999	0.0
Belgium	0.8317	0.1599	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0000	0.0
Bulgaria	0.0000	0.0000	0.0	0.9999	0.0000	0.0000	0.0000	0.0	0.0000	0.0
Croatia	0.0000	0.9999	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0000	0.0
Cyprus	0.0000	0.6718	0.0	0.1707	0.0289	0.0000	0.0000	0.0	0.1286	0.0
Czechia	0.0000	0.0000	0.0	0.0000	0.0000	0.0000	0.9998	0.0	0.0000	0.0
Denmark	0.0550	0.9448	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0000	0.0
Estonia	0.0000	0.0000	0.0	0.0000	0.0000	0.9997	0.0000	0.0	0.0000	0.0
Finland	0.0000	0.9701	0.0	0.0000	0.0000	0.0297	0.0000	0.0	0.0000	0.0
France	0.0795	0.0000	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.9205	0.0
Germany	0.0000	0.9999	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0000	0.0
Greece	0.0000	0.2681	0.0	0.5280	0.2037	0.0000	0.0000	0.0	0.0000	0.0
Hungary	0.0000	0.0000	0.0	0.0940	0.8740	0.0000	0.0000	0.0	0.0318	0.0
Ireland	0.9003	0.0629	0.0	0.0000	0.0000	0.0366	0.0000	0.0	0.0000	0.0
Italy	0.0000	0.0000	0.0	0.9999	0.0000	0.0000	0.0000	0.0	0.0000	0.0
Latvia	0.0000	0.0000	0.0	0.0000	0.0000	0.9997	0.0000	0.0	0.0000	0.0
Lithuania	0.0000	0.0000	0.0	0.2230	0.1210	0.2824	0.0214	0.0	0.3462	0.0
Luxembourg	0.0233	0.9761	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0000	0.0
Malta	0.0000	0.0837	0.0	0.2409	0.6753	0.0000	0.0000	0.0	0.0000	0.0
Netherlands	0.0511	0.9488	0.0	0.0000	0.0000	0.0000	0.0000	0.0	0.0000	0.0
Poland	0.0000	0.0000	0.0	0.9966	0.0000	0.0000	0.0000	0.0	0.0000	0.0
Portugal	0.0000	0.0000	0.0	0.0000	0.9999	0.0000	0.0000	0.0	0.0000	0.0
Romania	0.0000	0.0000	0.0	0.9999	0.0000	0.0000	0.0000	0.0	0.0000	0.0
Slovakia	0.0000	0.0000	0.0	0.0210	0.0193	0.0000	0.0461	0.0	0.9135	0.0
Slovenia	0.0000	0.0000	0.0	0.9997	0.0000	0.0000	0.0000	0.0	0.0000	0.0
Spain	0.0000	0.0000	0.0	0.0000	0.9998	0.0000	0.0000	0.0	0.0000	0.0
Sweden	0.9852	0.0000	0.0	0.0000	0.0000	0.0143	0.0000	0.0	0.0000	0.0

In [150]:
```
import scipy.spatial as sp
import scipy.cluster.hierarchy as hc
linkage = hc.linkage(overview_topic_probs, method='average', metric='cosine')
overview_similarities = sp.distance.squareform(sp.distance.pdist(overview_topic_probs.values, metric='cosine'))
```

In [151]:
```
plt.figure(figsize=(12, 8))
sns.clustermap(1-overview_similarities,
           xticklabels=overview_topic_probs.index,
           yticklabels=overview_topic_probs.index,
            row_linkage=linkage, col_linkage=linkage)
plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```



In [152]:
```
necp_processed
```

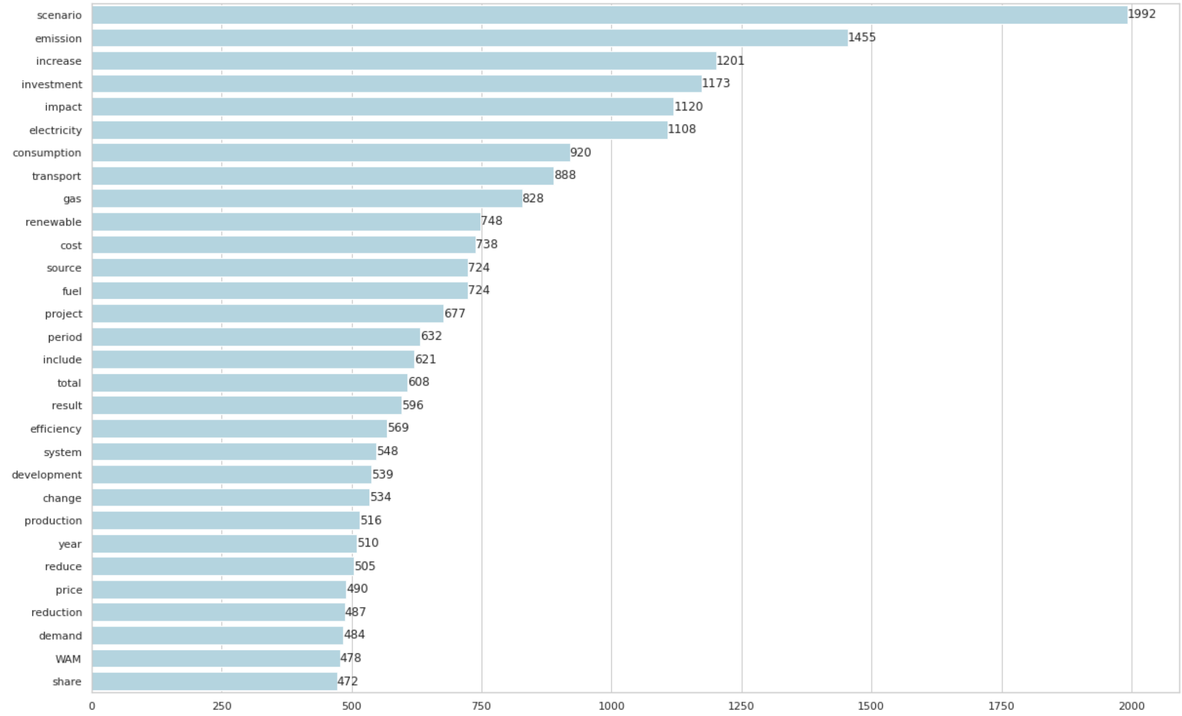
Out[152]:

	country	file_name	subsection	energy_union_dimension	text	necp_lem
0	Austria	at_final_necp_main_en_0.pdf	Overview and Process for Establishing the Plan	NaN	Integrated National Energy and Climate Plan fo...	[overview, DEVELOPMENT, PROCESS, Sumn F
1	Austria	at_final_necp_main_en_0.pdf	National Objectives and Targets	Decarbonisation	Integrated National Energy and Climate Plan fo...	[Decarbonisation, greenhouse, gas, emission
2	Austria	at_final_necp_main_en_0.pdf	National Objectives and Targets	Energy efficiency	2.2. Dimension 2: Energy efficiency i. indic...	[efficiency, indicative, contribution, EU,
3	Austria	at_final_necp_main_en_0.pdf	National Objectives and Targets	Energy security	2.3. Dimension 3: Security of energy supply ...	[security, supply, regard, diversify, sourc
4	Austria	at_final_necp_main_en_0.pdf	National Objectives and Targets	Internal market	Integrated National Energy and Climate Plan fo...	[internal, market, interconnectivity, elect
...	...	...	...	...	...	
454	Slovakia	sk_final_necp_main_en_0.pdf	Current Situation and Reference Projections	Energy efficiency	4.3 Dimension: energy efficiency i. Current p...	[efficiency, current, primary, final, consun
455	Slovakia	sk_final_necp_main_en_0.pdf	Current Situation and Reference Projections	Energy security	4.4. Dimension: energy security i. Current en...	[security, current, mix, domestic, resource
456	Slovakia	sk_final_necp_main_en_0.pdf	Current Situation and Reference Projections	Internal market	4.5. Dimension: internal energy market 4.5.1. ...	[internal, market, electricity, interconnec
457	Slovakia	sk_final_necp_main_en_0.pdf	Current Situation and Reference Projections	R&I and Competitiveness	4.6. Dimension: Research, innovation and compe...	[research, innovation, competitiveness, curr
458	Slovakia	sk_final_necp_main_en_0.pdf	Impact Assessment of Planned Policies and Meas...	NaN	5. IMPACT ASSESSMENT OF PLANNED POLICIES AND ...	[impact, planned, impact, describe, system, (

453 rows × 6 columns

subsection: Impact Assessment of Planned Policies and Measures

In [153]:
```
impact_docs = necp_processed[(necp_processed['subsection'] == "Impact Assessment of Planned Policies and Measures")]["necp_lemmas"]
impact_counter = Counter(impact_docs.sum()).most_common(30)
plot_counter(impact_counter)
plt.show()
```



In [154]:
```
impact_docs = impact_docs.apply(lambda doc: [lemma for lemma in doc if not (lemma in ['emission', 'scenario'])])
```

In [155]:
```
impact_dictionary = Dictionary(impact_docs)
impact_dictionary.filter_extremes(no_below=2, no_above=1.0)
impact_encoded_docs = impact_docs.apply(impact_dictionary.doc2bow)
```

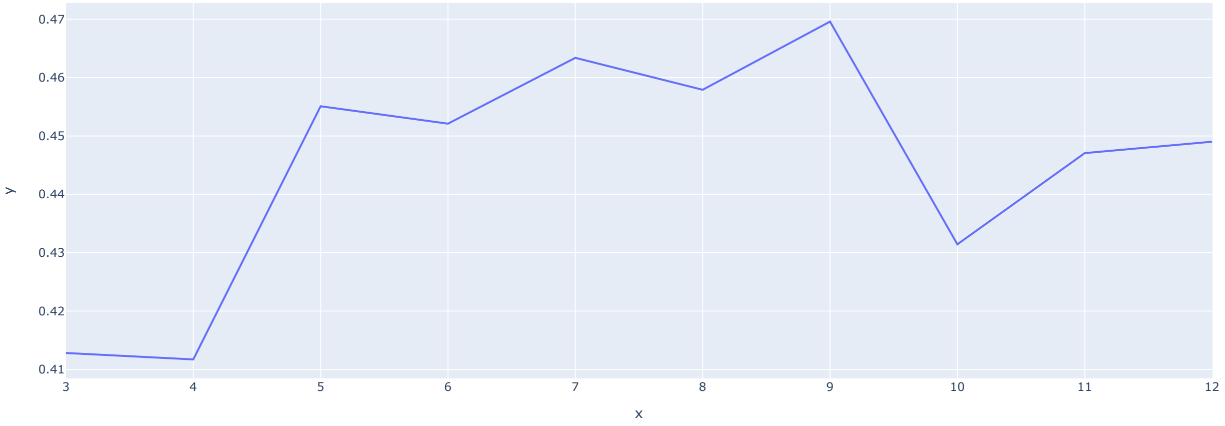
In [156]:
```
impact_models = []
for topics_number in tqdm(range(3, 13)):
    lda = LdaMulticore(impact_encoded_docs, num_topics=topics_number, passes=8, iterations=100, random_state=123)
    impact_models.append(lda)
```

```
  0%|          | 0/10 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning:

Calling np.sum(generator) is deprecated, and in the future will give a different result. Use np.sum(np.fromiter(generator)) or the python sum builtin instead.

100%|██████████| 10/10 [00:40<00:00,  4.08s/it]
```

In [157]:
```
impact_cvs = []
for model in tqdm(impact_models):
    cm = CoherenceModel(model,texts=impact_docs, dictionary=impact_dictionary)
    c_v = cm.get_coherence()
    impact_cvs.append(c_v)
```

```
100%|██████████| 10/10 [00:34<00:00,  3.50s/it]
```

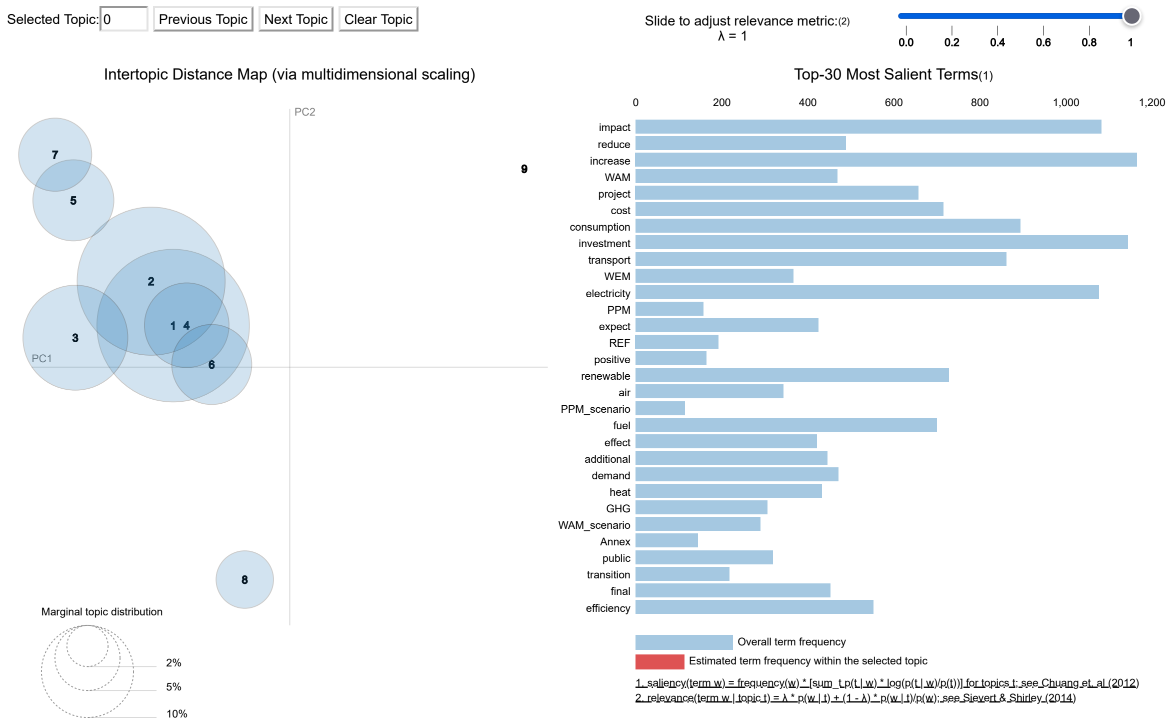
In [158]:
```
px.line(x=range(3, 13), y=impact_cvs)
```



```
In [159]: vis = pyLDAvis.gensim_models.prepare(impact_models[6], impact_encoded_docs, dictionary=impact_dictionary)
          vis
```

```
/usr/local/lib/python3.7/dist-packages/pyLDAvis/_prepare.py:247: FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
```

Out[159]:



```
In [160]: for idx, topic in impact_models[6].show_topics(formatted=False, num_words=15):
              print('Topic: {} \nWords: {}'.format(idx, [impact_dictionary[int(w[0])] for w in topic]))
```

```
Topic: 0
Words: ['transport', 'renewable', 'consumption', 'increase', 'electricity', 'GHG', 'demand', 'total', 'WAM', 'share', 'impact', 'project', 'res', 'period', 'gas']
Topic: 1
Words: ['investment', 'efficiency', 'increase', 'impact', 'project', 'economy', 'fund', 'carbon', 'financing', 'transition', 'reduction', 'new', 'change', 'achieve', 'sup
t']
Topic: 2
Words: ['project', 'expect', 'electricity', 'increase', 'investment', 'consumption', 'heat', 'impact', 'pump', 'heat_pump', 'period', 'technology', 'result', 'WEM', 'cost
Topic: 3
Words: ['impact', 'reduce', 'increase', 'positive', 'air', 'transport', 'effect', 'additional', 'incl', 'negative', 'need', 'public', 'fuel', 'environmental', 'assess']
Topic: 4
Words: ['investment', 'electricity', 'increase', 'gas', 'impact', 'WAM', 'source', 'include', 'period', 'cost', 'transport', 'total', 'system', 'development', 'consumptio
Topic: 5
Words: ['consumption', 'increase', 'investment', 'impact', 'renewable', 'transport', 'compare', 'term', 'carbon', 'WAM', 'reduction', 'electricity', 'final', 'estimate',
oduction']
Topic: 6
Words: ['consumption', 'renewable', 'electricity', 'impact', 'increase', 'investment', 'gas', 'source', 'demand', 'efficiency', 'fuel', 'WAM', 'share', 'cost', 'transport
Topic: 7
Words: ['fuel', 'impact', 'source', 'increase', 'electricity', 'gas', 'consumption', 'plant', 'result', 'production', 'price', 'transport', 'REF', 'include', 'model']
Topic: 8
Words: ['cost', 'electricity', 'investment', 'PPM', 'WEM', 'PPM_scenario', 'transport', 'increase', 'fuel', 'gas', 'demand', 'generation', 'result', 'change', 'WEM_scenar
']
```

- Topic 0: transport, renewable, consumption, GHG (Greenhouse Gases), biofuel
- Topic 1: investment, efficiency, increase, impact, financing
- Topic 2: project, expect, electricity, heat pump, pam
- Topic 3: impact, reduce, increase, positive
- Topic 4: investment, electricity, increase, WAM, gas, programme
- Topic 5: consumption, increase, term, carbon, INECP (International Nonproliferation Export Control Program)
- Topic 6: 0%
- Topic 7: fuel, impact, source, REF (Renewable Energy Foundation), plant, Annex
- Topic 8: cost, PPM, WEM, investment

```
In [161]: impact_corpus_model = impact_models[6][impact_encoded_docs]
```

```
In [162]: impact_metainfo = necp_processed[(necp_processed['subsection'] == "Impact Assessment of Planned Policies and Measures")]
          res_len = len(impact_metainfo)
          res = np.zeros((res_len, 9))
```

```
In [163]: for i, doc in enumerate(impact_corpus_model):
            for topic in doc:
              res[i][topic[0]] = np.round(topic[1], 4)
```

```
In [164]: impact_modeling_results = pd.concat([impact_metainfo.reset_index(drop=True), pd.DataFrame(res)], axis=1)
          impact_topic_probs = impact_modeling_results.groupby("country").mean().loc[:,[0, 1, 2, 3, 4, 5, 6, 7, 8]]
```

In [165]:
```
impact_topic_probs
```

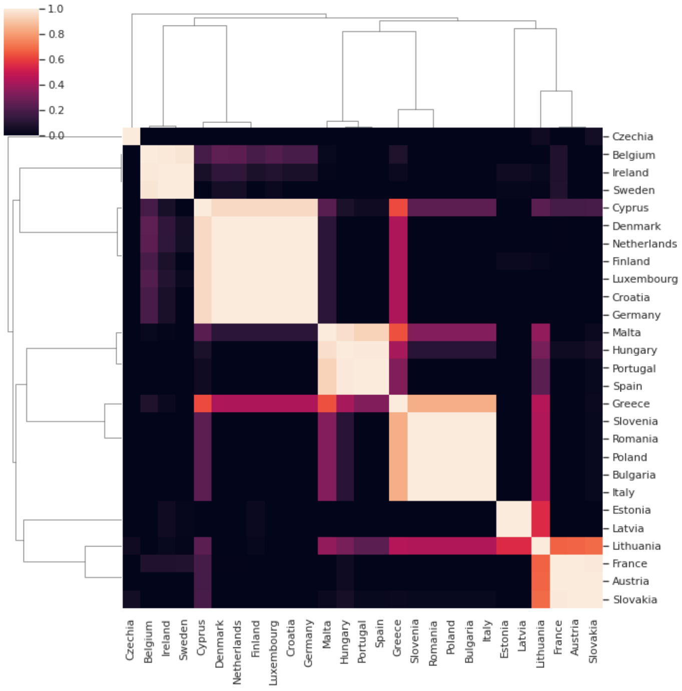
Out[165]:

	0	1	2	3	4	5	6	7	8
country									
Austria	0.0000	0.0000	0.0000	0.0000	0.9530	0.0466	0.0	0.0000	0.0000
Belgium	0.0000	0.0000	0.0000	0.0000	0.0000	0.9998	0.0	0.0000	0.0000
Bulgaria	0.0000	0.0600	0.0000	0.0144	0.2891	0.0000	0.0	0.6364	0.0000
Croatia	0.0000	0.0000	0.0000	0.0000	0.9992	0.0000	0.0	0.0000	0.0000
Cyprus	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	0.0000	0.9999
Czechia	0.0000	0.0000	0.0000	0.0000	0.9998	0.0000	0.0	0.0000	0.0000
Denmark	0.0000	0.0000	0.0000	0.0000	0.9991	0.0000	0.0	0.0000	0.0000
Estonia	0.0000	0.0000	0.0000	0.9998	0.0000	0.0000	0.0	0.0000	0.0000
Finland	0.0000	0.0000	0.0000	0.0000	0.9398	0.0000	0.0	0.0530	0.0000
France	0.0000	0.0000	0.0000	0.0000	0.0000	0.9997	0.0	0.0000	0.0000
Germany	0.0000	0.0000	0.0000	0.0000	0.9998	0.0000	0.0	0.0000	0.0000
Greece	0.0000	0.9750	0.0000	0.0000	0.0214	0.0000	0.0	0.0000	0.0000
Hungary	0.9997	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	0.0000	0.0000
Ireland	0.3117	0.0000	0.0920	0.0000	0.5665	0.0000	0.0	0.0000	0.0297
Italy	0.0000	0.0000	0.0000	0.0000	0.0420	0.5687	0.0	0.3891	0.0000
Latvia	0.7445	0.0000	0.0000	0.0179	0.1186	0.0000	0.0	0.1188	0.0000
Lithuania	0.9998	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	0.0000	0.0000
Luxembourg	0.9992	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	0.0000	0.0000
Malta	0.0000	0.0000	0.9999	0.0000	0.0000	0.0000	0.0	0.0000	0.0000
Netherlands	0.0000	0.0000	0.0000	0.0000	0.9996	0.0000	0.0	0.0000	0.0000
Poland	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	1.0000	0.0000
Portugal	0.0000	0.9999	0.0000	0.0000	0.0000	0.0000	0.0	0.0000	0.0000
Romania	0.0000	0.0000	0.0000	0.0000	0.9261	0.0636	0.0	0.0000	0.0000
Slovakia	0.0000	0.0000	0.0000	0.0000	0.8318	0.0000	0.0	0.1680	0.0000
Slovenia	0.0391	0.0000	0.0000	0.0000	0.7314	0.0121	0.0	0.2172	0.0000
Spain	0.0000	0.0000	0.0000	0.0000	0.0000	0.0906	0.0	0.9061	0.0000
Sweden	0.0000	0.0000	0.0000	0.0000	0.9996	0.0000	0.0	0.0000	0.0000

In [166]:
```
import scipy.spatial as sp
import scipy.cluster.hierarchy as hc
linkage = hc.linkage(overview_topic_probs, method='average', metric='cosine')
impact_similarities = sp.distance.squareform(sp.distance.pdist(overview_topic_probs.values, metric='cosine'))
```

In [167]:
```
plt.figure(figsize=(12, 8))
sns.clustermap(1-impact_similarities,
           xticklabels=overview_topic_probs.index,
           yticklabels=overview_topic_probs.index,
            row_linkage=linkage, col_linkage=linkage)
plt.show()
```

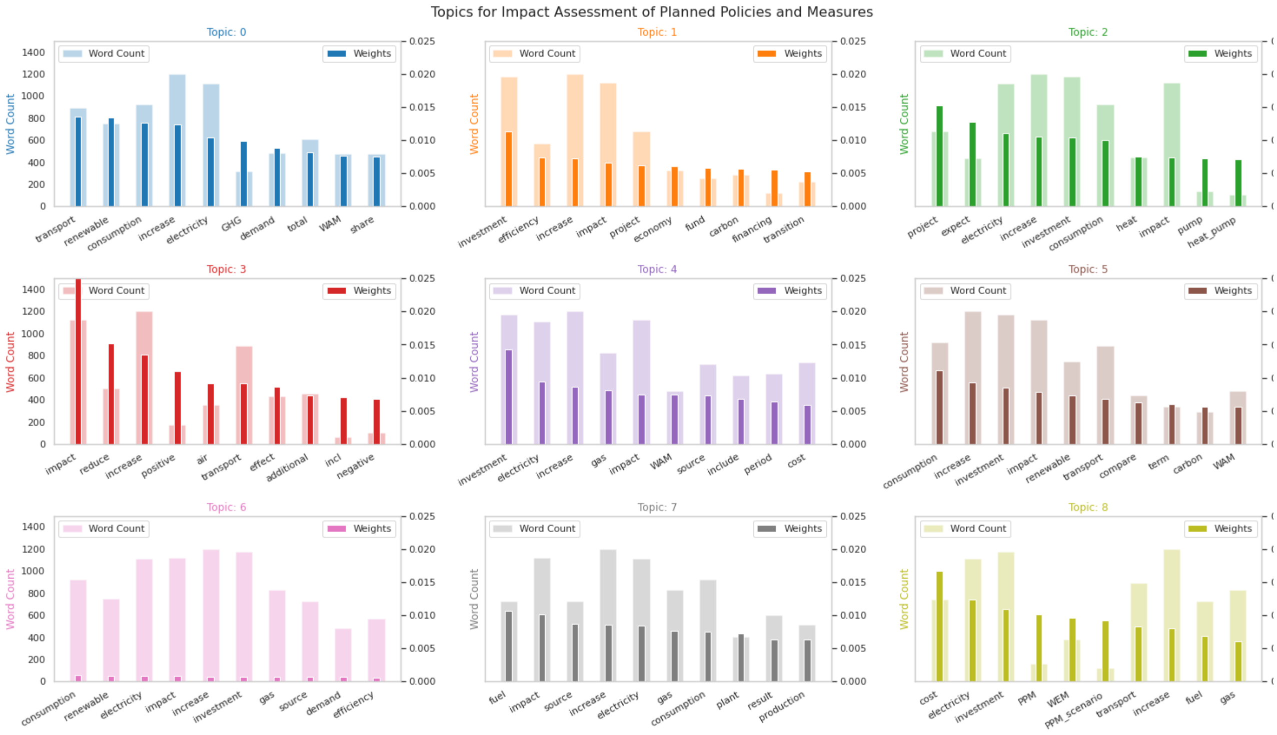
```
<Figure size 864x576 with 0 Axes>
```



In [168]:

```
from matplotlib import colors
topics = impact_models[6].show_topics(formatted=False)
counter = Counter(impact_docs.sum())

out = []
for i, topic in topics:
    for word, weight in topic:
        word = impact_dictionary[int(word)]
        out.append([word, i , weight, counter[word]])

df = pd.DataFrame(out, columns=['word', 'topic_id', 'importance', 'word_count'])

fig, axes = plt.subplots(3, 3, figsize=(21,12), sharey=True)
cols = [color for name, color in colors.TABLEAU_COLORS.items()]

for i, ax in enumerate(axes.flatten()):
    ax.bar(x='word', height="word_count", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.5, alpha=0.3, label='Word Count')
    ax_twin = ax.twinx()
    ax_twin.bar(x='word', height="importance", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.2, label='Weights')
    ax.set_ylabel('Word Count', color=cols[i])
    ax_twin.set_ylim(0, 0.025); ax.set_ylim(0, 1500)
    ax.set_title('Topic: ' + str(i), color=cols[i], fontsize=12)
    ax.tick_params(axis='y', left=False)
    ax.set_xticklabels(df.loc[df.topic_id==i, 'word'], rotation=30, horizontalalignment= 'right')
    ax.legend(loc='upper left'); ax_twin.legend(loc='upper right')
    ax.grid(False)
    ax_twin.grid(False)
fig.suptitle('Topics for Impact Assessment of Planned Policies and Measures', fontsize=16)
fig.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:24: UserWarning:

FixedFormatter should only be used together with FixedLocator
```


Topics for Impact Assessment of Planned Policies and Measures
Topic: 0
Topic: 1
Topic: 2
Topic: 3
Topic: 4
Topic: 5
Topic: 6
Topic: 7
Topic: 8
Word Count
Weights
transport
renewable
consumption
increase
electricity
GHG
demand
total
WAM
share
investment
efficiency
increase
impact
project
economy
fund
carbon
financing
transition
project
expect
electricity
increase
investment
consumption
heat
impact
pump
heat_pump
impact
reduce
increase
positive
air
transport
effect
additional
incl
negative
investment
electricity
increase
gas
impact
WAM
source
include
period
cost
consumption
increase
investment
impact
renewable
transport
compare
term
carbon
WAM
consumption
renewable
electricity
impact
increase
investment
gas
source
demand
efficiency
fuel
impact
source
increase
electricity
gas
consumption
plant
result
production
cost
electricity
investment
PPM
WEM
PPM_scenario
transport
increase
fuel
gas