In [1]:
```
import pandas as pd

pd.set_option('display.max_rows', 100)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', 500)
```

In [2]:
```
news_path = [redacted]
news_df = pd.read_parquet(news_path, engine='pyarrow')

print(f'Sample contains {news_df.shape[0]:,.0f} news articles')
news_df.head(2)
```

```
Sample contains 199,838 news articles
```

Out[2]:

	url	date	language	title	
0	http://auckland.scoop.co.nz/2020/01/aut-boosts-ai-expertise-with-new-ailab/	2020-01-28	en	auckland.scoop.co.nz » AUT boosts AI expertise with new AiLab	\n\nauckland.scoop.co.nz » AUT boosts AI experti \n\nTweet\nAUT boosts AI expertise with new AiLab\n
1	http://en.people.cn/n3/2021/0318/c90000-9830122.html	2021-03-18	en	Artificial intelligence improves parking efficiency in Chinese cities - People's Daily Online	\n\nArtific China\nBusiness\nMilitary\nWorld\nSociety\nCulture\n

In [3]:
```
news_df.shape
```

Out[3]:
```
(199838, 5)
```

In [4]:
```
!pip install nltk
```

```
Collecting nltk
  Downloading nltk-3.8.1-py3-none-any.whl (1.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 18.2 MB/s eta 0:00:0000:010:01
Requirement already satisfied: joblib in /opt/conda/lib/python3.7/site-packages (from nltk) (1.2.0)
Collecting regex>=2021.8.3
  Downloading regex-2022.10.31-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (757 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 757.1/757.1 kB 63.4 MB/s eta 0:00:00
Requirement already satisfied: click in /opt/conda/lib/python3.7/site-packages (from nltk) (8.1.3)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packages (from nltk) (4.64.1)
Requirement already satisfied: importlib-metadata in /opt/conda/lib/python3.7/site-packages (from click->n
ltk) (6.0.0)
Requirement already satisfied: typing-extensions>=3.6.4 in /opt/conda/lib/python3.7/site-packages (from im
portlib-metadata->click->nltk) (4.4.0)
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.7/site-packages (from importlib-metadat
a->click->nltk) (3.11.0)
Installing collected packages: regex, nltk
Successfully installed nltk-3.8.1 regex-2022.10.31
```

In [5]:
```
!pip install spacy
```

```
Collecting spacy
  Downloading spacy-3.5.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (6.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.5/6.5 MB 48.7 MB/s eta 0:00:0000:0100:01
Collecting preshed<3.1.0,>=3.0.2
  Downloading preshed-3.0.8-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manyli
nux2014_x86_64.whl (126 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 126.6/126.6 kB 18.7 MB/s eta 0:00:00
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.7/site-packages (from spacy) (23.
0)
Requirement already satisfied: typing-extensions<4.5.0,>=3.7.4.1 in /opt/conda/lib/python3.7/site-packages
(from spacy) (4.4.0)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /opt/conda/lib/python3.7/site-packages (from spa
cy) (2.28.2)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4 in /opt/conda/lib/python3.7/site-pack
ages (from spacy) (1.10.4)
Collecting pathy>=0.10.0
  Downloading pathy-0.10.1-py3-none-any.whl (48 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.9/48.9 kB 8.7 MB/s eta 0:00:00
Collecting spacy-legacy<3.1.0,>=3.0.11
  Downloading spacy_legacy-3.0.12-py2.py3-none-any.whl (29 kB)
Collecting srsly<3.0.0,>=2.4.3
  Downloading srsly-2.4.6-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (490 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 490.9/490.9 kB 47.9 MB/s eta 0:00:00
Requirement already satisfied: typer<0.8.0,>=0.3.0 in /opt/conda/lib/python3.7/site-packages (from spacy)
(0.7.0)
Collecting cymem<2.1.0,>=2.0.2
  Downloading cymem-2.0.7-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (36 kB)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /opt/conda/lib/python3.7/site-packages (from sp
acy) (6.3.0)
Collecting murmurhash<1.1.0,>=0.28.0
  Downloading murmurhash-1.0.9-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.man
ylinux2014_x86_64.whl (21 kB)
Collecting wasabi<1.2.0,>=0.9.1
  Downloading wasabi-1.1.1-py3-none-any.whl (27 kB)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages (from spacy) (3.1.2)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.7/site-packages (from spacy) (66.1.1)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /opt/conda/lib/python3.7/site-packages (from spacy)
(4.64.1)
Collecting catalogue<2.1.0,>=2.0.6
  Downloading catalogue-2.0.8-py3-none-any.whl (17 kB)
Collecting spacy-loggers<2.0.0,>=1.0.0
  Downloading spacy_loggers-1.0.4-py3-none-any.whl (11 kB)
Collecting langcodes<4.0.0,>=3.2.0
  Downloading langcodes-3.3.0-py3-none-any.whl (181 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 181.6/181.6 kB 27.7 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.15.0 in /opt/conda/lib/python3.7/site-packages (from spacy) (1.21.
6)
Collecting thinc<8.2.0,>=8.1.0
  Downloading thinc-8.1.9-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (912 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 912.1/912.1 kB 53.8 MB/s eta 0:00:00
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.7/site-packages (from catalogue<2.1.0,>
=2.0.6->spacy) (3.11.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.7/site-packages (from reque
sts<3.0.0,>=2.13.0->spacy) (1.26.14)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.7/site-packages (from requests<3.0.
0,>=2.13.0->spacy) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/site-packages (from requests
<3.0.0,>=2.13.0->spacy) (2022.12.7)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python3.7/site-packages (from re
quests<3.0.0,>=2.13.0->spacy) (2.1.1)
Collecting confection<1.0.0,>=0.0.1
  Downloading confection-0.0.4-py3-none-any.whl (32 kB)
Collecting blis<0.8.0,>=0.7.8
  Downloading blis-0.7.9-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (10.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10.2/10.2 MB 82.8 MB/s eta 0:00:0000:0100:01
Requirement already satisfied: click<9.0.0,>=7.1.1 in /opt/conda/lib/python3.7/site-packages (from typer<
0.8.0,>=0.3.0->spacy) (8.1.3)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-packages (from jinja2->spa
cy) (2.1.2)
Requirement already satisfied: importlib-metadata in /opt/conda/lib/python3.7/site-packages (from click<9.
0.0,>=7.1.1->typer<0.8.0,>=0.3.0->spacy) (6.0.0)
Installing collected packages: cymem, wasabi, spacy-loggers, spacy-legacy, murmurhash, langcodes, catalogu
e, blis, srsly, preshed, confection, thinc, pathy, spacy
Successfully installed blis-0.7.9 catalogue-2.0.8 confection-0.0.4 cymem-2.0.7 langcodes-3.3.0 murmurhash-
1.0.9 pathy-0.10.1 preshed-3.0.8 spacy-3.5.0 spacy-legacy-3.0.12 spacy-loggers-1.0.4 srsly-2.4.6 thinc-8.
1.9 wasabi-1.1.1
```

In [6]:

```
import pandas as pd
import nltk
from nltk import word_tokenize, pos_tag, ne_chunk
from nltk.chunk import conlltags2tree, tree2conlltags
from nltk.tree import Tree
import spacy
```

```
from spacy import displacy
from collections import Counter
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
```

Removing all the news data which are not in English language

In [7]:
```
news_df = news_df[news_df["language"]=='en']
news_df.reset_index(inplace = True)
```

Cleaning the Data

In [8]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/jupyter/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[8]: True

In [9]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /home/jupyter/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

Out[9]: True

In [10]:
```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /home/jupyter/nltk_data...
```

Out[10]: True

In [11]:
```
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to /home/jupyter/nltk_data...
```

Out[11]: True

In [ ]:
```
import pandas as pd
import re
import string
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

def clean_text(text):
    # Lowercasing
    text = text.lower()

    # Removing HTML tags
    text = re.sub(r'<.*?>', '', text)

    # Removing URLs
    text = re.sub(r'https?://\S+', '', text)

    # Removing Punctuation
    text = re.sub(f'[{string.punctuation}]', '', text)

    #Removing any other symbols
    text= re.sub(r'[^\w^\s]','',text)

    # Replace newlines and tabs with spaces
    text = re.sub(r'\n|\t', ' ', text)

    # Remove web crawl remnants
    text = re.sub(r'http\S+', '', text)

    # Remove any remaining non-printable characters and extra whitespace
    text = re.sub(r'[^\x00-\x7F]+', '', text).strip()

    # Removing Stopwords
    stop_words = set(stopwords.words("english"))
    words = word_tokenize(text)
    words = [word for word in words if word not in stop_words]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
```

```
    # Removing Numbers
    lemmatized_words = [word for word in lemmatized_words if not word.isdigit()]

    # Removing White Spaces
    lemmatized_words = [word.strip() for word in lemmatized_words]

    # Joining the words back into a single string
    cleaned_text = ' '.join(lemmatized_words)

    return cleaned_text

# Cleaning the text column
news_df['text_cleaned'] = news_df['text'].apply(clean_text)
```

In [ ]:
```
news_df["text"][0]
```

In [ ]:
```
news_df["text_cleaned"][0]
```

In [ ]:
```
news_df["title"][0]
```

In [ ]:
```
news_df.head(2)
```

It is seen that the text starts with title. Hence, we cna use text for the analysis.

Saved the cleaned dataset so that we do not have to run the cleaning data part everytime we run the notebook.

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

```
Mounted at /content/drive
```

In [ ]:
```
news_df.to_parquet('news_df_cleaned.parquet')
!cp news_df_cleaned.parquet "drive/My Drive/"
```

In [ ]:
```

```

Reading the cleaned data set from the google drive

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

In [1]:
```
import os
import pandas as pd

os.chdir('/content/drive/My Drive')
news_df = pd.read_parquet('/content/drive/My Drive/news_df_cleaned.parquet',engine='pyarrow')
```

In [ ]:
```
news_df.head(2)
```

Out[ ]:

	index	url	date	language	title	text	text_cleane
0	0	http://auckland.scoop.co.nz/2020/01/aut-boosts...	2020-01-28	en	auckland.scoop.co.nz » AUT boosts AI expertise...	\n\nauckland.scoop.co.nz » AUT boosts AI exper...	aucklandscoopcon aut boost expertise new a.
1	1	http://en.people.cn/n3/2021/0318/c90000-983012...	2021-03-18	en	Artificial intelligence improves parking effic...	\n\nArtificial intelligence improves parking e...	artifici intelligenc improves parkin effic.

In [ ]:
```
news_df.shape
```

Out[ ]:
```
(199838, 7)
```

Applying LDA on Tweets Data

In [10]:
```
import pandas as pd

news_df=pd.read_parquet("news_df_cleaned.parquet")
news_df.shape
```

Out[10]: (199838, 7)

In [11]:
```
!pip install pyLDAvis
```

```
Requirement already satisfied: pyLDAvis in /opt/conda/lib/python3.7/site-packages (3.3.1)
Requirement already satisfied: numexpr in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (2.8.4)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (3.1.2)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (1.7.3)
Requirement already satisfied: numpy>=1.20.0 in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (1.
21.6)
Requirement already satisfied: future in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (0.18.3)
Requirement already satisfied: pandas>=1.2.0 in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (1.
3.5)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (1.
0.2)
Requirement already satisfied: joblib in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (1.2.0)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (66.1.
1)
Requirement already satisfied: funcy in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (1.18)
Requirement already satisfied: gensim in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (4.2.0)
Requirement already satisfied: sklearn in /opt/conda/lib/python3.7/site-packages (from pyLDAvis) (0.0.post
1)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/site-packages (from pand
as>=1.2.0->pyLDAvis) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-packages (from pandas>=1.2.0-
>pyLDAvis) (2022.7.1)
Requirement already satisfied: smart-open>=1.8.1 in /opt/conda/lib/python3.7/site-packages (from gensim->p
yLDAvis) (6.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-packages (from jinja2->pyL
DAvis) (2.1.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from scikit
-learn->pyLDAvis) (3.1.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from python-dateutil>=
2.7.3->pandas>=1.2.0->pyLDAvis) (1.16.0)
```

In [7]:
```
import time
import math
import re
#from textblob import TextBlob
import pandas as pd

import nltk as nltk
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer

import string


import gensim
from gensim import corpora, models
from gensim.models.ldamulticore import LdaMulticore

# import pyLDAvis.gensim
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis
pyLDAvis.enable_notebook()
```

In [12]:
```
news_list = news_df['text_cleaned'].tolist()
#del news_df

news_clean = [doc.split() for doc in news_list]
#del news_list
```

In [13]:
```
# Creating the term dictionary of our corpus, where every unique term is assigned an index.

dictionary = corpora.Dictionary(news_clean)

# Converting list of documents (corpus) into Document Term Matrix using dictionary prepared above.

%time doc_term_matrix = [dictionary.doc2bow(doc) for doc in news_clean]
```

```
CPU times: user 1min 57s, sys: 4.22 s, total: 2min 2s
Wall time: 2min 1s
```

In [18]:
```
%%time

#Using multicore LDA
num_topics = 5
iterations = 50
passes = 20
workers = 10
eval_every = None
```

```
ldamodel = LdaMulticore(corpus=doc_term_matrix,
                        id2word=dictionary,
                        eta='auto',
                        num_topics=num_topics,
                        iterations=iterations,
                        passes=passes,
                        eval_every=eval_every,
                        workers = workers)
```

```
CPU times: user 26min 49s, sys: 3min 17s, total: 30min 6s
Wall time: 33min 26s
```

In [19]:
```
print(*ldamodel.print_topics(num_topics=5, num_words=7), sep='\n')
doc_topics = [ldamodel.get_document_topics(bow) for bow in doc_term_matrix]

doc_topics[0:5]
```

```
(0, '0.061*"market" + 0.019*"report" + 0.018*"intelligence" + 0.017*"artificial" + 0.016*"global" + 0.015
*"analysis" + 0.013*"growth"')
(1, '0.008*"ai" + 0.008*"news" + 0.006*"u" + 0.006*"new" + 0.003*"video" + 0.003*"said" + 0.003*"technolog
y"')
(2, '0.008*"ai" + 0.008*"stock" + 0.008*"market" + 0.008*"company" + 0.008*"share" + 0.006*"news" + 0.004
*"digi"')
(3, '0.020*"ai" + 0.010*"data" + 0.006*"technology" + 0.005*"company" + 0.005*"medium" + 0.005*"solution"
+ 0.005*"gray"')
(4, '0.012*"u" + 0.011*"service" + 0.010*"news" + 0.009*"product" + 0.009*"release" + 0.009*"technology" +
0.009*"business"')
```

Out[19]:
```
[[(1, 0.52183163), (2, 0.47700143)],
 [(0, 0.1355158), (1, 0.6795303), (3, 0.18420073)],
 [(0, 0.99876606)],
 [(0, 0.14734314), (1, 0.8037689), (2, 0.048427682)],
 [(0, 0.037364718), (1, 0.90428936), (3, 0.04500656), (4, 0.012997646)]]
```

In [20]:
```
#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.8 for _, val in sublist) for sublist in doc_topics]
print("Min 80%: ", result.count(True))

#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.7 for _, val in sublist) for sublist in doc_topics]
print("Min 70%: ", result.count(True))

#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.6 for _, val in sublist) for sublist in doc_topics]
print("Min 60%: ", result.count(True))
```

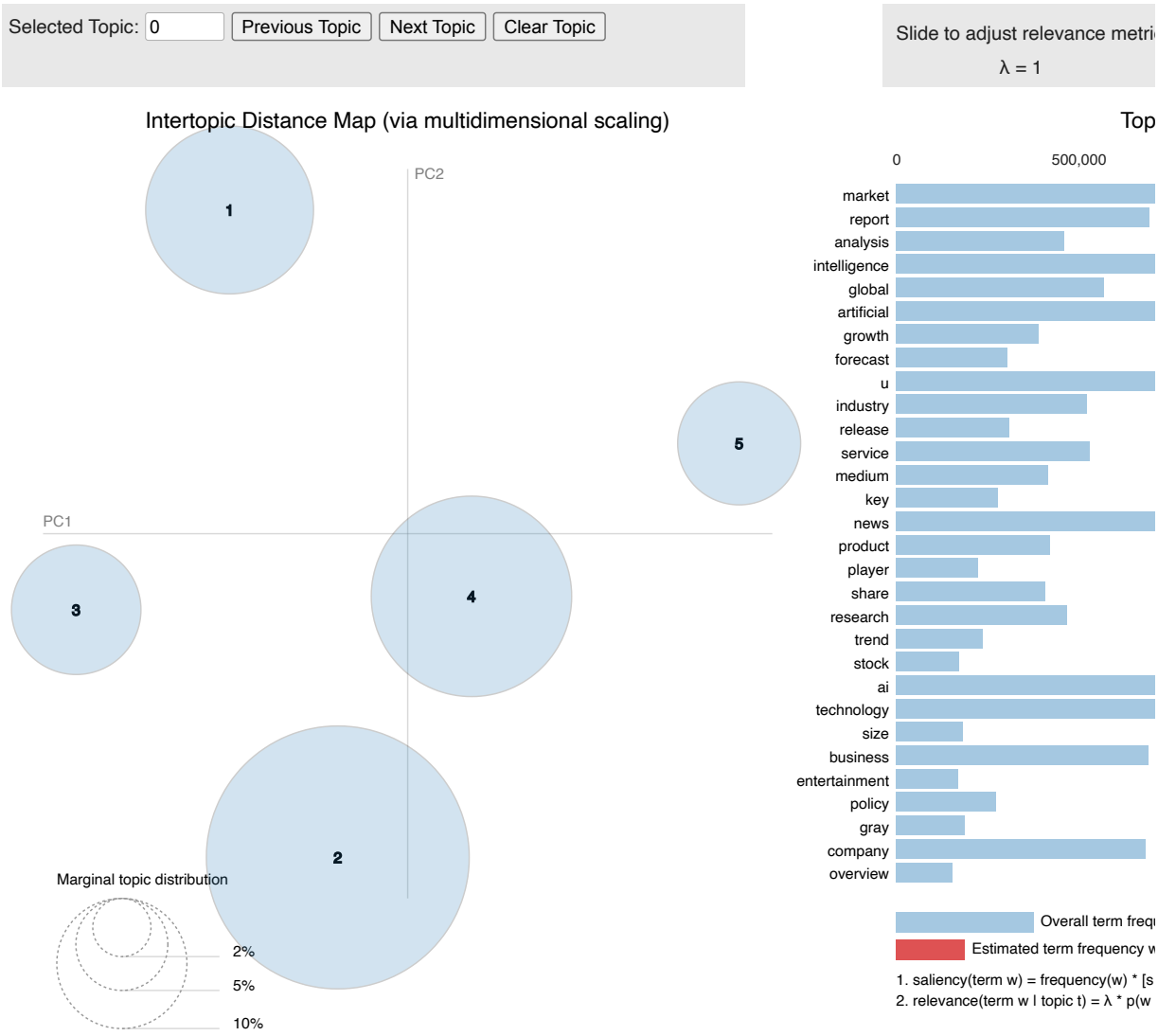
```
Min 80%:  132099
Min 70%:  150864
Min 60%:  168246
```

In [21]:
```
%%time

lda_display = gensimvis.prepare(ldamodel, doc_term_matrix, dictionary, sort_topics=False, mds='mmds')
pyLDAvis.display(lda_display)
```

```
/opt/conda/lib/python3.7/site-packages/pyLDAvis/_prepare.py:247: FutureWarning: In a future version of pan
das all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  by='saliency', ascending=False).head(R).drop('saliency', 1)
CPU times: user 10min 17s, sys: 8min 43s, total: 19min
Wall time: 4min 49s
```

Out[21]:



In [23]:

```
%%time

#Using multicore LDA
num_topics = 4
iterations = 50
passes = 20
workers = 10
eval_every = None

ldamodel = LdaMulticore(corpus=doc_term_matrix,
                        id2word=dictionary,
                        eta='auto',
                        num_topics=num_topics,
                        iterations=iterations,
                        passes=passes,
                        eval_every=eval_every,
                        workers = workers)
```

```
CPU times: user 24min 30s, sys: 3min 5s, total: 27min 36s
Wall time: 30min 9s
```

In [24]:

```
print(*ldamodel.print_topics(num_topics=4, num_words=7), sep='\n')

doc_topics = [ldamodel.get_document_topics(bow) for bow in doc_term_matrix]
doc_topics[0:5]
```

```
(0, '0.044*"market" + 0.014*"report" + 0.013*"intelligence" + 0.013*"artificial" + 0.012*"global" + 0.011
*"analysis" + 0.010*"ai"')
(1, '0.010*"ai" + 0.009*"news" + 0.006*"u" + 0.006*"new" + 0.004*"data" + 0.003*"technology" + 0.003*"busi
ness"')
(2, '0.007*"ai" + 0.006*"stock" + 0.005*"new" + 0.005*"day" + 0.005*"share" + 0.004*"company" + 0.003*"new
s"')
(3, '0.019*"ai" + 0.008*"data" + 0.007*"technology" + 0.006*"company" + 0.006*"medium" + 0.005*"solution"
+ 0.005*"group"')
```

Out[24]:
```
[[(0, 0.026651883), (1, 0.5445222), (2, 0.41923565)],
 [(0, 0.10950805), (1, 0.7548037), (3, 0.13519858)],
 [(0, 0.9988023)],
 [(0, 0.13877565), (1, 0.6790719), (2, 0.18185319)],
 [(0, 0.034465984), (1, 0.9646432)]]
```

In [25]:
```
#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.8 for _, val in sublist) for sublist in doc_topics]
print("Min 80%: ", result.count(True))

#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.7 for _, val in sublist) for sublist in doc_topics]
print("Min 70%: ", result.count(True))

#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.6 for _, val in sublist) for sublist in doc_topics]
print("Min 60%: ", result.count(True))
```

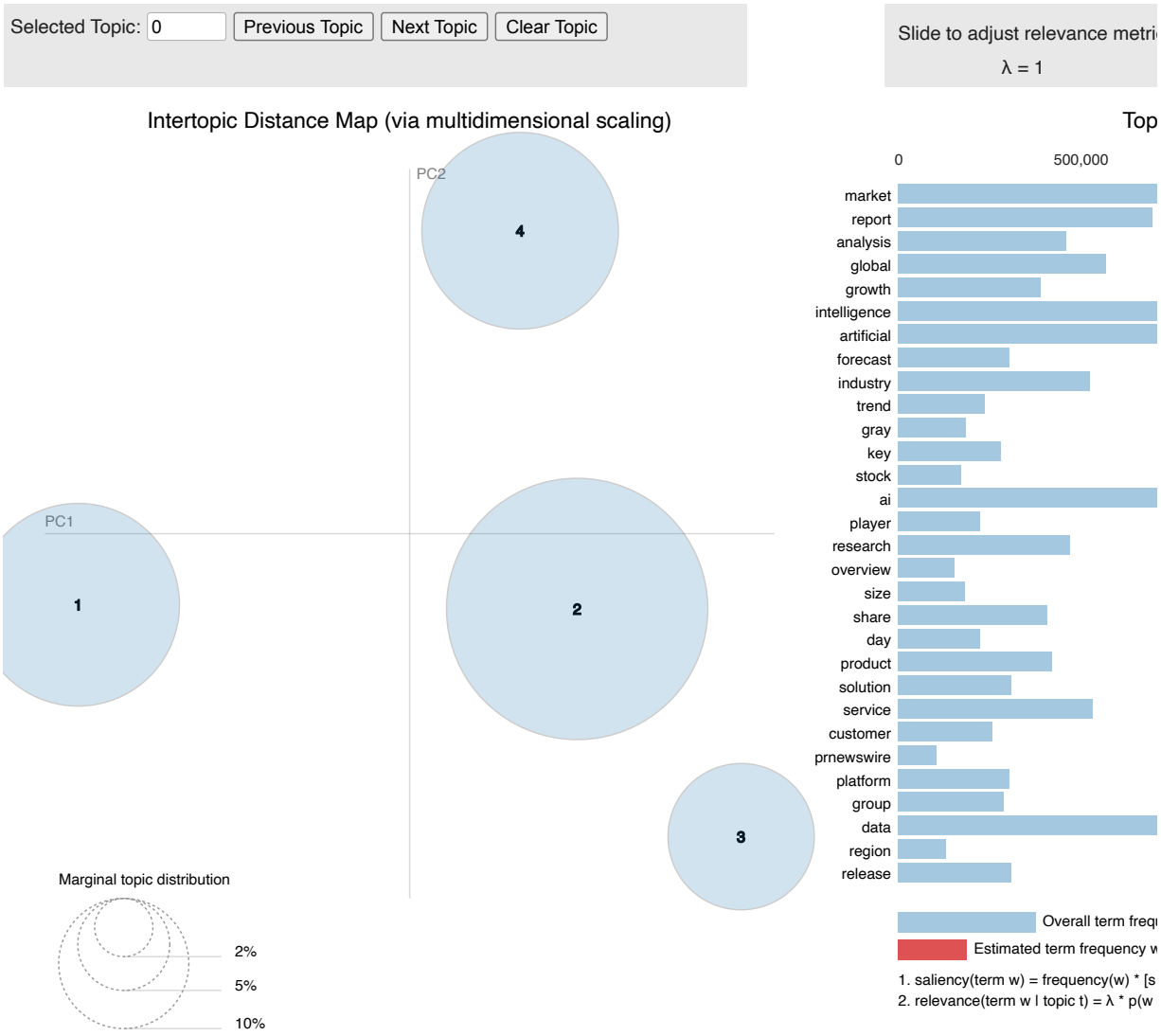
```
Min 80%:  137726
Min 70%:  158992
Min 60%:  175871
```

In [26]:
```
%%time

lda_display = gensimvis.prepare(ldamodel, doc_term_matrix, dictionary, sort_topics=False, mds='mmds')
pyLDAvis.display(lda_display)
```

```
/opt/conda/lib/python3.7/site-packages/pyLDAvis/_prepare.py:247: FutureWarning: In a future version of pan
das all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  by='saliency', ascending=False).head(R).drop('saliency', 1)
CPU times: user 8min 55s, sys: 7min, total: 15min 56s
Wall time: 4min 30s
```

Out[26]:



In [ ]:

In [27]:
```
%%time

#Using multicore LDA
num_topics = 6
iterations =50
passes = 20
workers = 10
eval_every = None

ldamodel = LdaMulticore(corpus=doc_term_matrix,
                        id2word=dictionary,
                        eta='auto',
                        num_topics=num_topics,
                        iterations=iterations,
                        passes=passes,
                        eval_every=eval_every,
                        workers = workers)
```

```
CPU times: user 28min 36s, sys: 4min 49s, total: 33min 25s
Wall time: 40min 47s
```

In [28]:
```
print(*ldamodel.print_topics(num_topics=6, num_words=7), sep='\n')
print("\n")
doc_topics = [ldamodel.get_document_topics(bow) for bow in doc_term_matrix]
doc_topics[0:5]
```

```
(0, '0.011*"stock" + 0.010*"ai" + 0.009*"company" + 0.009*"market" + 0.008*"share" + 0.006*"fund" + 0.006
*"news"')
(1, '0.019*"ai" + 0.007*"data" + 0.007*"gray" + 0.006*"medium" + 0.006*"technology" + 0.006*"group" + 0.00
6*"solution"')
(2, '0.008*"ai" + 0.007*"news" + 0.005*"new" + 0.005*"technology" + 0.004*"people" + 0.004*"business" + 0.
004*"service"')
(3, '0.013*"u" + 0.009*"news" + 0.006*"new" + 0.005*"ai" + 0.005*"said" + 0.004*"video" + 0.004*"ago"')
(4, '0.062*"market" + 0.020*"report" + 0.018*"intelligence" + 0.017*"artificial" + 0.016*"global" + 0.015
*"analysis" + 0.013*"growth"')
(5, '0.015*"ai" + 0.011*"data" + 0.009*"news" + 0.007*"business" + 0.007*"u" + 0.005*"technology" + 0.005
*"company"')
```

Out[28]:
```
[[(0, 0.3224074), (2, 0.14592943), (3, 0.35972926), (5, 0.17130701)],
 [(2, 0.10624792), (3, 0.30752504), (4, 0.10887482), (5, 0.46931285)],
 [(4, 0.9987304)],
 [(0, 0.054370563),
  (2, 0.56585854),
  (3, 0.06790743),
  (4, 0.12275383),
  (5, 0.18892121)],
 [(2, 0.7773037), (4, 0.02889154), (5, 0.19296038)]]
```

In [29]:
```
#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.8 for _, val in sublist) for sublist in doc_topics]
print("Min 80%: ", result.count(True))

#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.7 for _, val in sublist) for sublist in doc_topics]
print("Min 70%: ", result.count(True))

#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.6 for _, val in sublist) for sublist in doc_topics]
print("Min 60%: ", result.count(True))
```

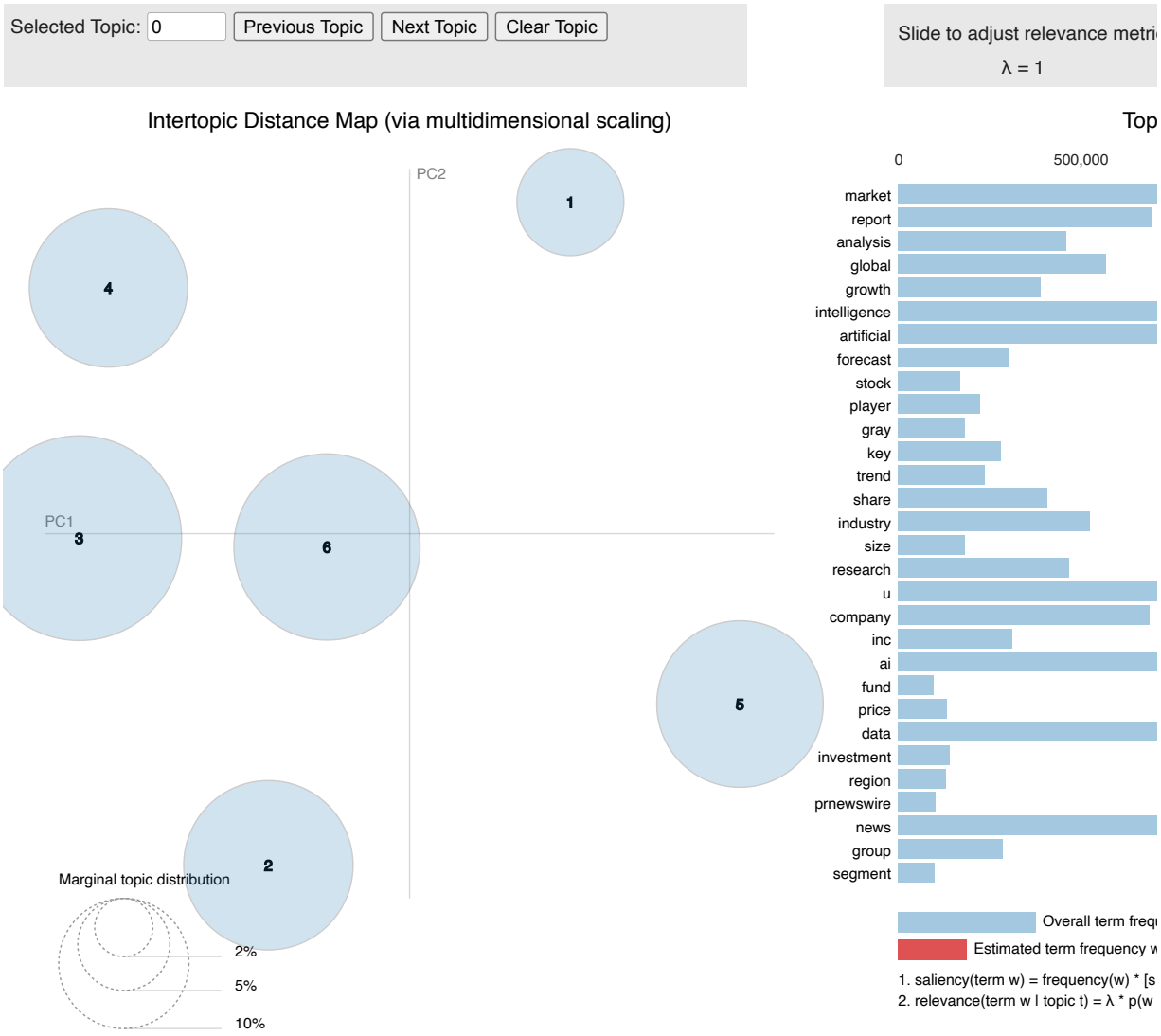
```
Min 80%:  109344
Min 70%:  134068
Min 60%:  157073
```

In [30]:
```
%%time

lda_display = gensimvis.prepare(ldamodel, doc_term_matrix, dictionary, sort_topics=False, mds='mmds')
pyLDAvis.display(lda_display)
```

```
/opt/conda/lib/python3.7/site-packages/pyLDAvis/_prepare.py:247: FutureWarning: In a future version of pan
das all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  by='saliency', ascending=False).head(R).drop('saliency', 1)
CPU times: user 13min 1s, sys: 11min 56s, total: 24min 58s
Wall time: 5min 19s
```

Out[30]:



In [31]:
```
%%time

#Using multicore LDA
num_topics = 7
iterations = 50
passes = 20
workers = 10
eval_every = None

ldamodel = LdaMulticore(corpus=doc_term_matrix,
                        id2word=dictionary,
                        eta='auto',
                        num_topics=num_topics,
                        iterations=iterations,
                        passes=passes,
                        eval_every=eval_every,
                        workers = workers)
```

```
CPU times: user 30min 47s, sys: 4min 52s, total: 35min 39s
Wall time: 48min 20s
```

In [32]:
```
print(*ldamodel.print_topics(num_topics=7, num_words=7), sep='\n')
print("\n")
doc_topics = [ldamodel.get_document_topics(bow) for bow in doc_term_matrix]
doc_topics[0:5]
```

```
(0, '0.012*"ai" + 0.012*"u" + 0.012*"data" + 0.006*"new" + 0.006*"learning" + 0.005*"news" + 0.005*"machin
e"')
(1, '0.065*"market" + 0.021*"report" + 0.018*"intelligence" + 0.017*"artificial" + 0.016*"global" + 0.015
*"analysis" + 0.013*"growth"')
(2, '0.013*"news" + 0.013*"service" + 0.012*"product" + 0.011*"business" + 0.010*"technology" + 0.010*"rel
ease" + 0.009*"medium"')
(3, '0.016*"ai" + 0.008*"gray" + 0.007*"group" + 0.006*"medium" + 0.006*"technology" + 0.006*"health" + 0.
005*"patient"')
(4, '0.008*"news" + 0.008*"ai" + 0.006*"new" + 0.005*"u" + 0.004*"said" + 0.004*"video" + 0.003*"say"')
(5, '0.020*"ai" + 0.009*"customer" + 0.007*"data" + 0.007*"solution" + 0.006*"platform" + 0.006*"technolog
y" + 0.006*"company"')
(6, '0.010*"ai" + 0.009*"stock" + 0.009*"news" + 0.008*"company" + 0.008*"market" + 0.007*"share" + 0.005
*"technology"')
```

Out[32]:
```
[[(0, 0.66503245), (4, 0.25411436), (5, 0.07979025)],
 [(0, 0.2564188), (1, 0.09358077), (4, 0.42524314), (5, 0.22396769)],
 [(1, 0.998701)],
 [(0, 0.41705796), (1, 0.11412788), (4, 0.46648505)],
 [(0, 0.9250731), (4, 0.07373756)]]
```

In [33]:
```
#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.8 for _, val in sublist) for sublist in doc_topics]
print("Min 80%: ", result.count(True))

#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.7 for _, val in sublist) for sublist in doc_topics]
print("Min 70%: ", result.count(True))

#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.6 for _, val in sublist) for sublist in doc_topics]
print("Min 60%: ", result.count(True))
```

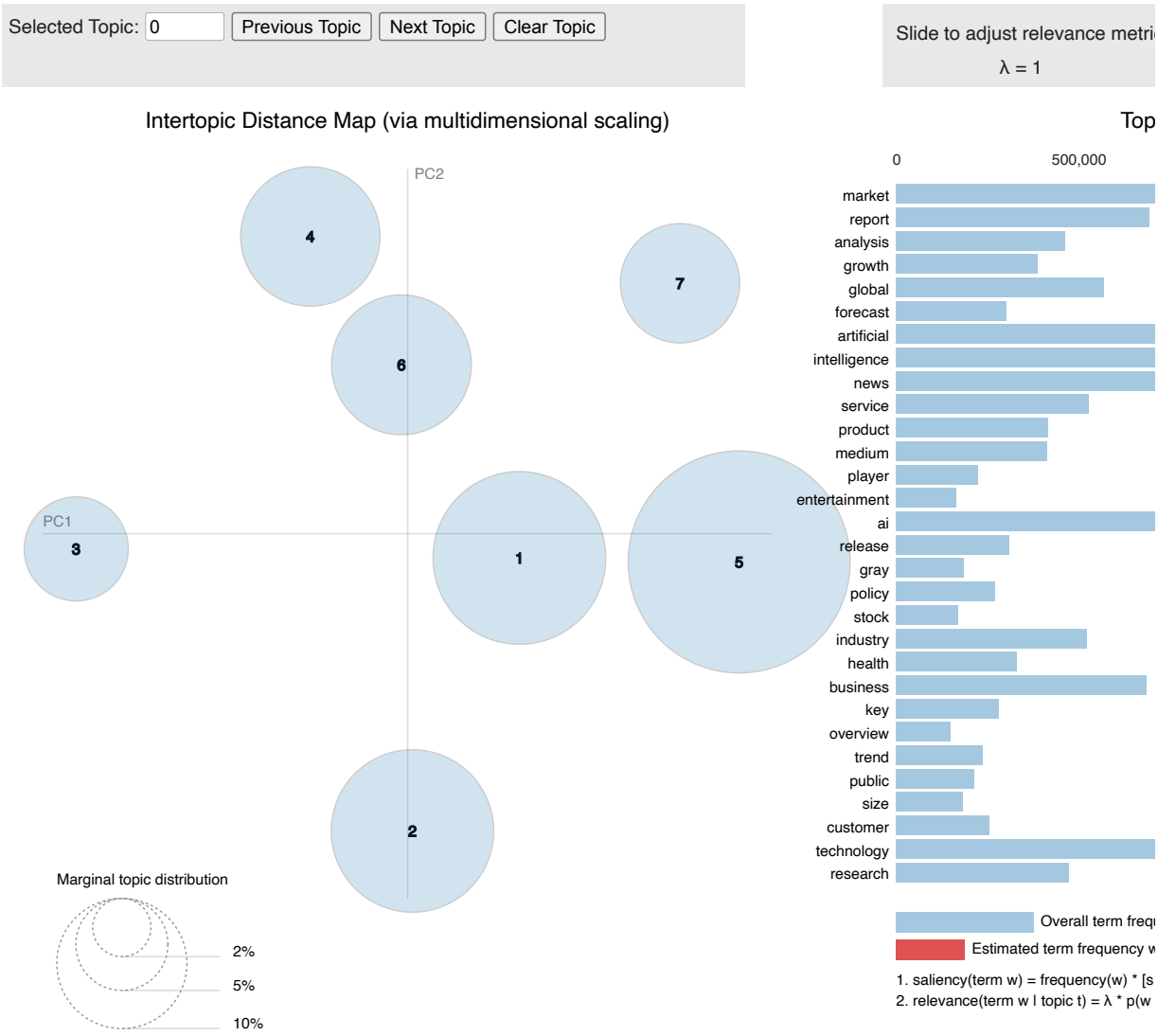
```
Min 80%:  98804
Min 70%:  125771
Min 60%:  151109
```

In [34]:
```
%%time

lda_display = gensimvis.prepare(ldamodel, doc_term_matrix, dictionary, sort_topics=False, mds='mmds')
pyLDAvis.display(lda_display)
```

```
/opt/conda/lib/python3.7/site-packages/pyLDAvis/_prepare.py:247: FutureWarning: In a future version of pan
das all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  by='saliency', ascending=False).head(R).drop('saliency', 1)
CPU times: user 16min 58s, sys: 17min 37s, total: 34min 36s
Wall time: 5min 29s
```

Out[34]:



In [35]:
```
%%time

#Using multicore LDA
num_topics = 10
iterations = 50
passes = 20
workers = 10
eval_every = None

ldamodel = LdaMulticore(corpus=doc_term_matrix,
                        id2word=dictionary,
                        eta='auto',
                        num_topics=num_topics,
                        iterations=iterations,
                        passes=passes,
                        eval_every=eval_every,
                        workers = workers)
```

```
CPU times: user 36min 14s, sys: 6min 31s, total: 42min 46s
Wall time: 1h 27min 26s
```

In [36]:
```
print(*ldamodel.print_topics(num_topics=7, num_words=10), sep='\n')
print("\n")
doc_topics = [ldamodel.get_document_topics(bow) for bow in doc_term_matrix]
doc_topics[0:5]
```

```
(9, '0.014*"service" + 0.014*"product" + 0.011*"business" + 0.011*"release" + 0.011*"technology" + 0.011
*"news" + 0.010*"medium" + 0.010*"entertainment" + 0.010*"overview" + 0.009*"resource"')
(8, '0.014*"digi" + 0.011*"communication" + 0.010*"market" + 0.010*"nv" + 0.008*"report" + 0.007*"share" +
0.007*"company" + 0.007*"fund" + 0.006*"transaction" + 0.005*"symbol"')
(3, '0.013*"health" + 0.012*"patient" + 0.008*"medical" + 0.007*"healthcare" + 0.007*"clinical" + 0.006*"a
i" + 0.006*"news" + 0.006*"cancer" + 0.006*"care" + 0.005*"research"')
(4, '0.018*"ai" + 0.013*"data" + 0.006*"learning" + 0.005*"business" + 0.005*"new" + 0.005*"technology" +
0.005*"machine" + 0.005*"u" + 0.004*"news" + 0.004*"model"')
(7, '0.022*"u" + 0.017*"news" + 0.007*"email" + 0.006*"weather" + 0.006*"business" + 0.005*"local" + 0.005
*"new" + 0.005*"video" + 0.005*"public" + 0.005*"republic"')
(6, '0.017*"stock" + 0.012*"ai" + 0.011*"market" + 0.010*"share" + 0.009*"company" + 0.008*"price" + 0.007
*"inc" + 0.007*"news" + 0.007*"trading" + 0.006*"trade"')
(5, '0.066*"market" + 0.021*"report" + 0.019*"intelligence" + 0.017*"artificial" + 0.017*"global" + 0.016
*"analysis" + 0.014*"growth" + 0.012*"ai" + 0.011*"industry" + 0.011*"research"')
```

Out[36]:
```
[[(0, 0.6336997), (1, 0.18721619), (3, 0.10770025), (8, 0.0702623)],
 [(0, 0.34118897),
  (1, 0.16053894),
  (4, 0.3639216),
  (5, 0.08160669),
  (7, 0.051828537)],
 [(3, 0.011853989), (5, 0.9869438)],
 [(0, 0.30726174),
  (1, 0.31547797),
  (3, 0.10015553),
  (4, 0.17180276),
  (5, 0.10474482)],
 [(1, 0.23785925), (3, 0.276584), (4, 0.44526595), (7, 0.03928395)]]
```

In [37]:
```
#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.8 for _, val in sublist) for sublist in doc_topics]
print("Min 80%: ", result.count(True))

#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.7 for _, val in sublist) for sublist in doc_topics]
print("Min 70%: ", result.count(True))

#Checking if each document has atleast 80% probability to belong to one single topic
result = [any(val >= 0.6 for _, val in sublist) for sublist in doc_topics]
print("Min 60%: ", result.count(True))
```

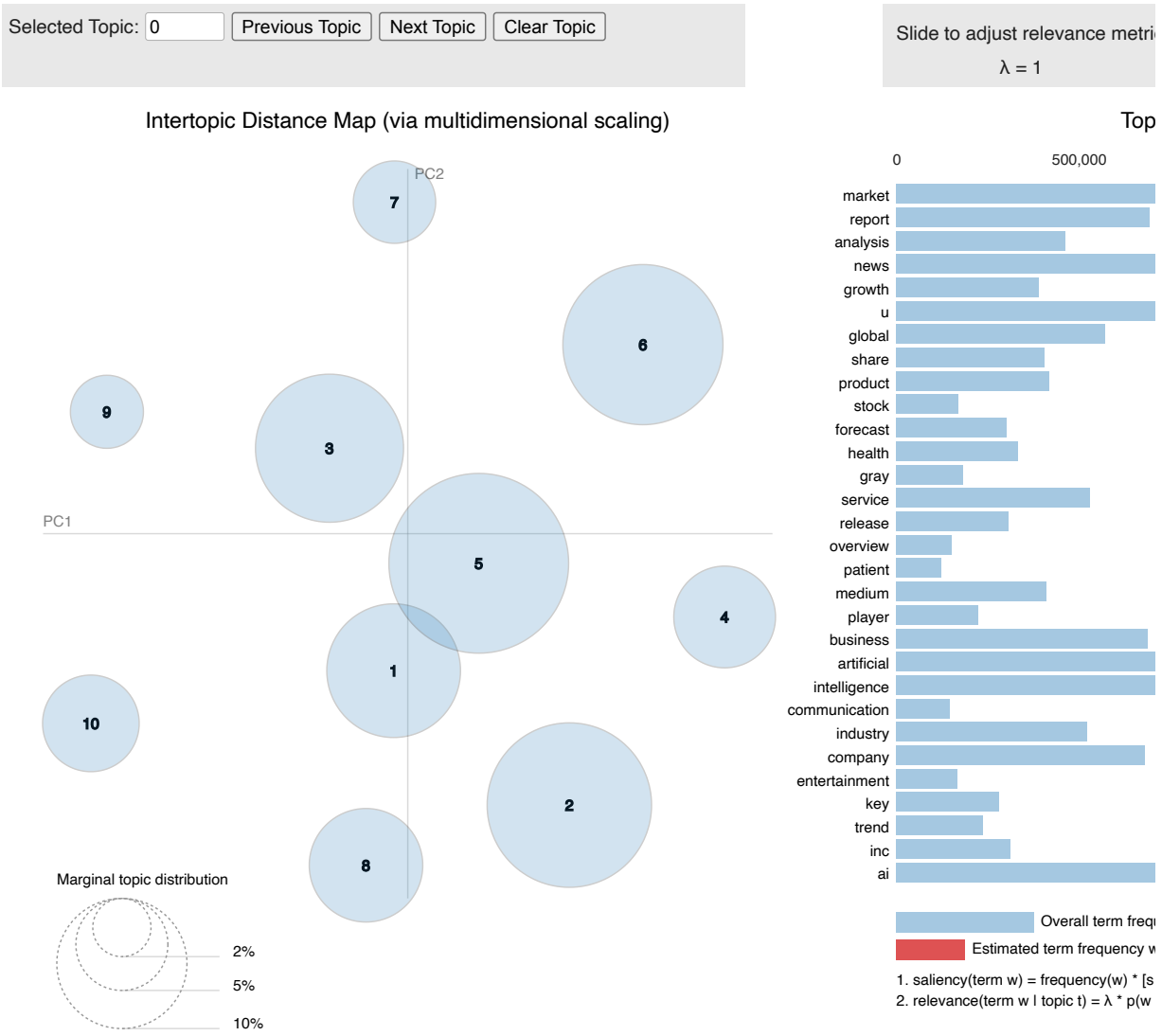
```
Min 80%:  89312
Min 70%:  114391
Min 60%:  141191
```

In [38]:
```
%%time

lda_display = gensimvis.prepare(ldamodel, doc_term_matrix, dictionary, sort_topics=False, mds='mmds')
pyLDAvis.display(lda_display)
```

```
/opt/conda/lib/python3.7/site-packages/pyLDAvis/_prepare.py:247: FutureWarning: In a future version of pan
das all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  by='saliency', ascending=False).head(R).drop('saliency', 1)
CPU times: user 32min 18s, sys: 37min 41s, total: 1h 9min 59s
Wall time: 6min 53s
```

Out[38]:



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Final Model for News Data

The final model chosen for news data is showcasing 4 topics.

- Topic 1 | ~17% of tokens
- Topic 2 | ~7% of tokens
- Topic 3 | ~8% of tokens
- Topic 4 | ~68% of tokens

Reasons to choose N=4:

- 86% of the data is being classified into one single topic with a confidence of atleast 60%. Hence, it is safe to say that there is minimal duplication.
- By looking at top words of each topic, we are able to figure out what the topics could be.

- The Intertopic Distance Map also shows that there are no cases of overlapping.

In [ ]: