Analyse automatique du narratif de corpus

`Shami THIRION SEN`

Lien vers le git du projet

Introduction

A la suite d'une première analyse du narratif du communisme au premier semestre, l'objectif du projet actuel est d'effectuer une analyse automatique du narratif avec les outils TAL. Les étapes principales pour la mener sont la constitution du corpus, un traitement efficace et une modélisation qui puisse « faire parler » le corpus. Sans une définition précise de ce qu'est le narratif du communisme, du point de vu TAL, nous partons avec l'intention de découvrir des termes associés au slogans communistes : *« ouvriers, travailleurs, peuple, égalité»* employés dans des journaux dits « communiste ». Nous effectuons également une étude de « corpus-contrastifs ». Plus précisément, nous allons catégoriser certains journaux comme « non-communistes » afin d'y observer la récurrences des termes et les contrastes, le cas échéant.

Les outils TAL employées dans cet objectif sont:
1. le topic modelling avec LDA gensim
2. le calcul de spécificité des corpus avec le package Textometry en R

Nous partons du principe que le narratif du communiste emploient d'avantages les termes liés aux peuples, à l'égalité et aux gouvernements. Cependant, n'ayant pas une telle hypothèse pour le corpus «non-communiste», nous ne cherchons pas de termes en particulier. En conséquence, ce travail est hypothétique et s'apparente davantage à une observation qu'à une définition stricte de notre objet de recherche, en particulier en ce qui concerne le corpus contrastif.

Constitution et descriptif du corpus

La constitution du corpus a été la première étape, et la plus complexe. La difficulté consistait, d'une part en définition de ce qu'est «non-communiste», et d'autre part en collecte des données suffisamment larges pour un traitement automatique. Pour la collecte et l'extraction nous avons choisi WayBack Machine qui permet un téléchargement automatique automatique, (quoique longue) des journaux choisis.

Corpus communiste

Afin d'observer des phénomènes linguistiques communiste la source principale choisie est le journal «Ganashakti», le pouvoir du peuple. Fondé par, dirigeant marxiste,CPI(M) du Bengale occidental, il a eu *« un impacte profond sur le narratif politique du bengale»*(Medium).

Il existe deux version du corpus ganashakti.com, version en ligne du journal et bangla.ganashakti.co.in, des scannes (de hautes qualités) du journal imprimé. Sur Internet Archives, les captures ganashakti.com renvoient des textes en anglais, alors que le site original est en bengali.L'extraction des images issues des `bangla.ganashakti.co.in` a permis d'extraire des données avec l'océrisation avec la bibliothèque pytesseract de Python. La quantité et la qualité de l'extraction sont impressionnantes, ça a été la source la plus simple en terme de collecte, plus de 250 000 tokens. Ganashakti étant le journal communiste le plus accessible, et le plus représentatif du discours communiste, il constitue notre unique corpus dans cette catégorie.

Corpus *non-communiste*

En absence d'une définition claire du « non-communiste », nous avons choisi les journaux les plus populaires au Bengal, Anandabazar Patrika et Bartamapatrika. Anandabazar Patrika, fondé en 1922 Historical Dictionary of the Bengalis ayant pour but un journalisme politiquement neutre. Quant au *Bartamapatrika*, pendant le régime communiste au Bengal il était connu pour ses opinions anticommunistes. C'est le deuxième journal le plus diffusé au Bengal, après Anandabazar Patrika. Comme les choix ne suivent pas une logique ferme de ce qu'est *anti* ou *non* communiste, nous nous basons sur ces deux jounaux pour consituer notre corpus «non-communiste».

Chronologie

Afin d'illustrer le narratif du communisme au bengal, la démarche idéal serait l'observation d'un corpus contemporain de l'époque où la partie communiste a gouverné l'État de Bengal de l'Ouest, entre entre 1977 et 2011, issu de la presse. Or, la digitalisation des journaux est un phénomène qui commence vers les années 2001 au Bengal. Afin d'obtenir un corpus plus large, bien que potentiellement moins représentatif, nous étendons notre recherche pour inclure toutes les données disponibles, de l'époque actuelle jusqu'au plus loin dans le passé possible.

Source et Extraction massive

L'outil wayback-machine-downloader (lien git) a rendu possible une tâche qui semblait impossible au début. Il est capable d'extraire TOUTES les horodatage (timestamps) disponibles sur WayBack Machine - Internet Archives. Néanmoins ce processus est long, à titre d'exemple : environ 20 heures pour l'extraction de Ganashakti.

Scripts et automatisation des tâches

La prochaine étape consiste en l'élaboration des scripts qui permettent l'extraction des corpus depuis Internet Archives.

1. Extraction «en vrac»
   Avec la commande `wayback_machine_downloader http://<nom_du_site>.com --all-timestamps` nous avons extrait des **15353** instances de html pour bangla_ganashakti, **139521** instances pour bartamanpatrika et **16965** instances pour anandabazar. Beaucoup de dossiers et sous-dossiers sont vides. Des raisons potentielles étant l'absence de contenu sur le site, ou le non-respect du délai pour les requêtes vers Internet Archives. Le script `save_images_iter.py` permet d'extaire les images issu de WayBack Machine pour bangla_ganashakti . `scrape.sh` contient les commandes bash pour l'extraction, mais il faut executer une ligne à la fois, les délais des traitement allant jusqu'à une vingtaines d'heures.

2. HTML Parsing
   Si l'extraction automatique du corpus représente un défi au niveau des capacités des machines, repérer et extraire le corpus à partir des fichiers HTML requiert l'identification des balises dont le contenu est utilisable comme corpus. Avec un parcours récursif de tous les fichiers html dans les répertoires, on récupère les balises `<p>` , qui sont souvent celles qui contiennent des informations textuelles. Mais d'autres balises contiennent également du texte. Une déchiffrage efficace des encodages et des polices (obsolète) aurait permis une extraction plus large de contenu disponible, dû au manque de temps, ça n'a pas pu être réalisé.

3. Preprocessing
   Comme souvent, le nettoyage et la mise en forme du corpus est une étape incontournable. Après un premier nettoyage lors de la lecture des corpus avec `bartamanpatrika.py` et `anadabazar.py` , nous en effectuons d'autres dans ce notebook. Pour `bangla_ganashakti` nous effectuons une océrisation avec `ocerise_banglaGanashakti.py` des images extraites. Après d'océrisation nous gardons que les caractères bengali `regex.sub(r'[^\p{Bengali}\s]` . Ensuite, pour chaque corpus, sauvegardé comme des fichiers `txt` , nous effectuons la lecture, la lemmatisation avec BnLemma et l' étiquetage morphosyntaxique en partie du discours avec la bibliothèque bnlp, nous choisissons que les étiquettes NC et NP, nom commun et nom propre respectivement. Nous éliminons ensuite les jours et les mois de l'année avec regex.

4. Calculs TAL
   Afin d'observer les termes qui représentent le corpus, nous avons choisi le Topic Modeling avec Gensim et le Calcul de spécificité avec R. Ces algorithmes nous permettent d'observer et de tester l'hypothèse de départ. Dans la partie suivante, nous effectuons les calculs nécessaires pour faire une interprétation pour chauqe corpus.

5. utils.py
   Pour alléger le notebook, les fonctions de lecture, preprocessing, et calculs de spécificités ont été exportées dans `utils.py` .

In [1]:
```
import numpy as np
import json
import glob
from pprint import pprint
from collections import defaultdict
import os, math
```

```
from utils import * # import des fonctions depuis le fichier utils

import gensim
import gensim.corpora as corpora
from gensim.corpora import Dictionary
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel

#visualisation
import pyLDAvis
import pyLDAvis.gensim

# outils TAL en bengali
import BnLemma as lm
from bnlp import BengaliPOS

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

In [2]:
```
# Bangla POS tagging
bn_pos = BengaliPOS()
from bnlp import BengaliCorpus as corpus
stopwords = corpus.stopwords[:20] + ['রি' + 'টি']
```

In [3]:
```
def gen_words(texts):
    single_string = [ ' '.join(line) for line in texts]
    data_words = [gensim.utils.simple_preprocess(text) for text in single_string]
    return data_words
```

In [4]:
```
## fonction pour créer des BIGRAMMES ET TRIGRAMMES avec Gensim
def make_bigrams(texts):
    # print(bigram[doc] for doc in texts)
    return ([bigram[doc] for doc in texts])

def make_trigrams(texts):
    return([trigram[bigram[doc]] for doc in texts])
```

On choisit le nombre de tokens - pour chaque corpus

In [5]:
```
num_tokens = 20000
```

BanglaGanashakti

Lecture du corpus et preprocessing

In [6]:
```
banglaGanashakti = "../corpus/txtFiles/banglaGanashakti.txt"
filtered_ganashakti = read_corpus(banglaGanashakti, num_tokens)
filtered_ganashakti = [list for list in filtered_ganashakti if len(list)>0] # old list_of_docs
```

Génération des bigrammes et trigrammes avec gensim

In [7]:
```
# generation
data_words  = gen_words(filtered_ganashakti)
# print(data_words[9][:200])

# copy_this
bigram_phrases = gensim.models.Phrases(filtered_ganashakti, min_count=5, threshold=50)
trigram_phrases = gensim.models.Phrases(bigram_phrases[filtered_ganashakti], threshold=50)

bigram = gensim.models.phrases.Phraser(bigram_phrases)
trigram = gensim.models.phrases.Phraser(trigram_phrases)

data_bigrams = make_bigrams(filtered_ganashakti)
data_bigrams_trigrams = make_trigrams(data_bigrams)

# data_bigrams_trigrams = make_bigrams_trigrams(filtered_ganashakti)
```

```
['দল', 'সময়', 'টগ', 'উদ', 'মরণ', 'গঠন', 'অঞ', 'চল']
```

In [8]:
```
pprint(data_bigrams_trigrams[:3])
```

```
[['স্কুল',
  'ঘটনা',
  'ছাত্র',
  'গভীর',
  'বেহালা',
  'রাস্তা',
  'গিয়ে',
  'বিপত্তি',
  'শামিল',
  'গ্রামবাসী',
  'বেহালা',
  'রাস্তা',
  'কাঁটা',
  'শুরু',
  'ইউস',
  'জেলা'],
 ['জরা'],
 ['জাল',
  'আন্দোলন',
  'রমজান',
  'রারোর',
  'ছরিমমিজি',
  'সুরা',
  'হাত',
  'যুবক',
  'বেসরকারি',
  'বোত্ুল',
  'তৈরির',
  'হাসপাতাল',
  'তদন্ত']]
```

Filtrage des termes pour conserver uniquement les termes dont les valeurs TF-IDF sont supérieures au seuil spécifié

In [9]:
```
### TF-IDF
id2word = Dictionary(data_bigrams_trigrams)
corpus = [id2word.doc2bow(text) for text in filtered_ganashakti]

word = id2word[[9][:1][0]]
print (word)
```

```
বেহালা
```

In [10]:
```
corpus=get_corpus(corpus,id2word)
```

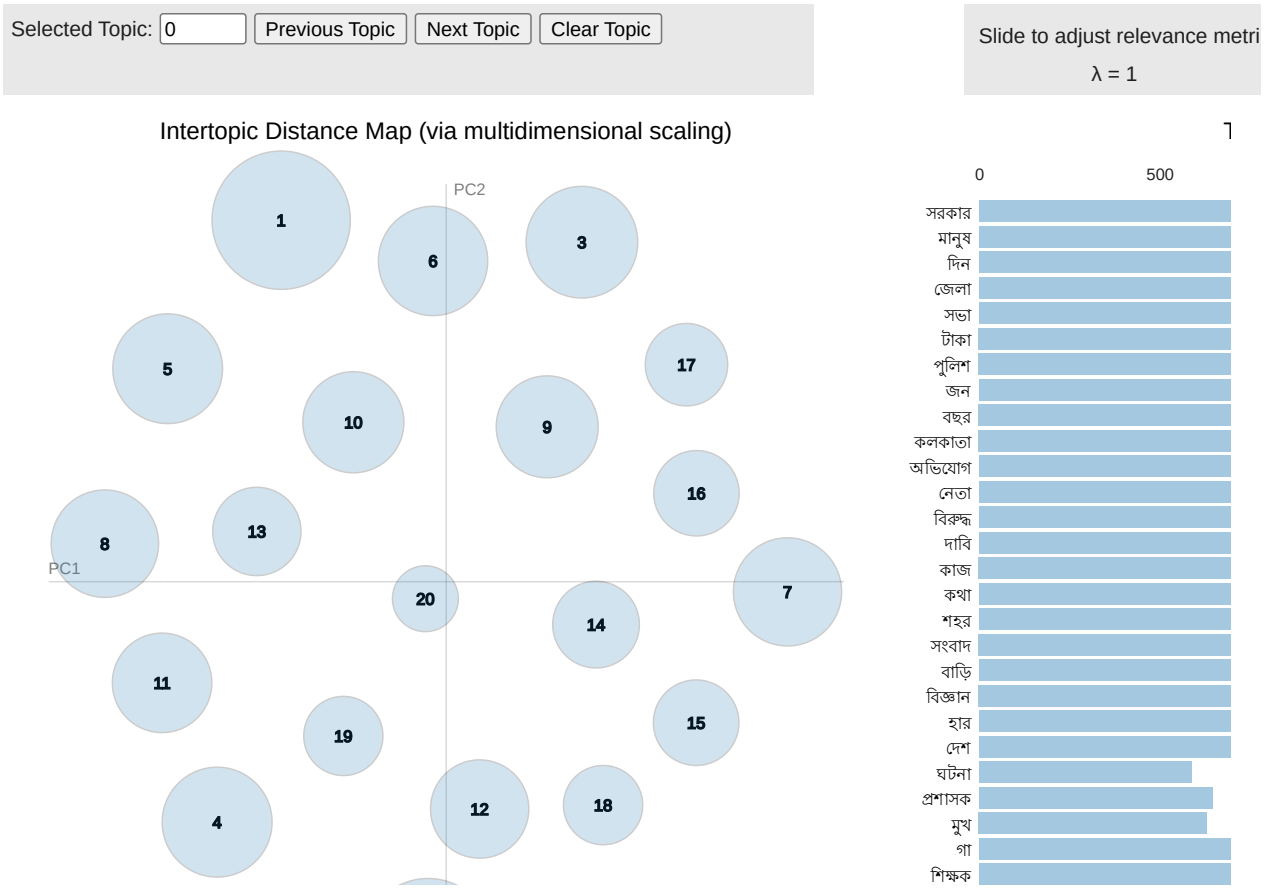
Création de modèle LDA et visualisation des *topics*

In [11]:
```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=20,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=200,
                                            passes=10,
                                            alpha="auto")
```

Création de modèle LDA et visualisation des *topics*

In [12]:
```
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word, mds="mmds", R=30)
vis
```

Out[12]:



In [13]:
```
# pyLDAvis.save_html(vis, '../topic_modeling_ganashaki.html')
```

Calcul spécificité

In [14]:
```
max_specificity, min_specificity = get_higest_lowest_specificity(filtered_ganashakti, 10)
```

In [15]:
```
print('Token les plus spécifiques:\n')
pprint(max_specificity)
print('\nToken les moins spécifiques:\n')
pprint(min_specificity)
```

```
Token les plus spécifiques:

{'কর্মচারী': 7.152601216831979,
 'গোটা': 5.247708031226533,
 'দান': 10.089226552041858,
 'ধারা': 5.797679820687194,
 'ফের': 6.701926734484466,
 'বিধি': 7.424882156047578,
 'ভাঙা': 5.642592340624519,
 'রায়': 5.840427693911302,
 'শাসক': 6.919229974339688,
 'সৃষ্টি': 5.190710911562063}

Token les moins spécifiques:

{'আন্দোলন': -3.756884777837276,
 'কৃষ্ণ': -3.6588851308850603,
 'টিম': -3.4371855815548513,
 'দের': -6.417412503116255,
 'পুলিশ': -4.1838163887554565,
 'প্রার্থীরা': -4.187354986603204,
 'ফেরার': -3.4371855815548513,
 'ব্যর্থ': -4.441768215309945,
 'সুনীল': -3.5022443187477,
 'হিজাব': -3.5022443187477}
```

Observation

Pour chaque corpus, nous avons gardé les mêmes hyperparamètres pour le modèles lda. Nous observons un par un le topic-modeling et les scores de spécificité pour chaque corpus.

Avec le modèle Gensim, nous avons extrait 20 sujets (topics). Avec le sujet 2, nous pouvons observer des termes : ***contre, travail, les nouvelles, organisation, le mouvement, ouvrier, poste, école, correspondre, nom, image, courir, ajoutée, les pauvres, Siliguri, correspondant à, la règle, le plan, comprendre, marié, Martyr, journaliste, Ville, bord, mixte, connecté, année, le bois, manches, enfant***. Les mots tel que ***mouvement ouvrier travail les pauvres*** peuvent être interprétée comme vocabulaire «communiste». Le topic 6, représenté par ***argent, demande, agriculteur, étudiant, vie, loi, prix, esprit, application, cas, honte, corporation, facture, classe, contre, étang, engrais, approuvé, vol*** traite les sujets des liés au «peuple» et agriculteur. Le sujet 3 semble ressembler des termes plus généraux concerant la politique tels que : ***personnes, le dirigeant, la ville, parce que, femme, question, le droit, le nombre, Congrès, essayer, Municipalité***. On inclus également le parti ***Congrès*** l'opposant principal du parti communiste.

Quant aux termes dites les plus spécifiques, obtenus avec le calcul de spécificité : ***employé, tous, don, section, fer, règle, cassé, verdict, création*** ont une légère nuance du vocabulaire des travailleurs.

Bartamanpatrika

Lecture du corpus et preprocessing

In [16]:
```
bartamanpatrika = "../corpus/txtFiles/bartamanpatrika.txt"
filtered_bartamanpatrika = read_corpus(bartamanpatrika, num_tokens)
filtered_bartamanpatrika = [list for list in filtered_bartamanpatrika if len(list)>0]
```

Génération des bigrammes et trigrammes avec gensim

In [17]:
```
# generation
data_words  = gen_words(filtered_bartamanpatrika)
# print(data_words[9][:200])

# copy_this
bigram_phrases = gensim.models.Phrases(filtered_bartamanpatrika, min_count=5, threshold=50)
trigram_phrases = gensim.models.Phrases(bigram_phrases[filtered_bartamanpatrika], threshold=50)

bigram = gensim.models.phrases.Phraser(bigram_phrases)
trigram = gensim.models.phrases.Phraser(trigram_phrases)

data_bigrams = make_bigrams(filtered_bartamanpatrika)
data_bigrams_trigrams = make_trigrams(data_bigrams)
```

In [18]:
```
print(data_bigrams_trigrams[:3])
```

```
[['কাপের', 'নাল', 'নাসা', 'বিরুদ্ধ', 'নামা', 'ঁাহন_বাগান'], ['সূত্র', 'বিত্ত', 'স্বাস্থ্য', 'ভ্রমণ', 'পরিকল্পনা', 'ধর্ম'], ['জোট', 'সিপাহি', 'দেওয়াল', 'হাত', 'চিহ্ন']]
```

Filtrage des termes pour conserver uniquement les termes dont les valeurs TF-IDF sont supérieures au seuil spécifié

In [19]:
```
### TF-IDF

id2word = Dictionary(data_bigrams_trigrams)
corpus = [id2word.doc2bow(text) for text in filtered_bartamanpatrika]

word = id2word[[9][:1][0]]
print (word)
```

```
ভ্রমণ
```

In [20]:
```
corpus=get_corpus(corpus,id2word)
```

Création de modèle LDA et visualisation des *topics*

In [21]:
```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=30,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha="auto")
```

In [22]:
```
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word, mds="mmds", R=30)
vis
```

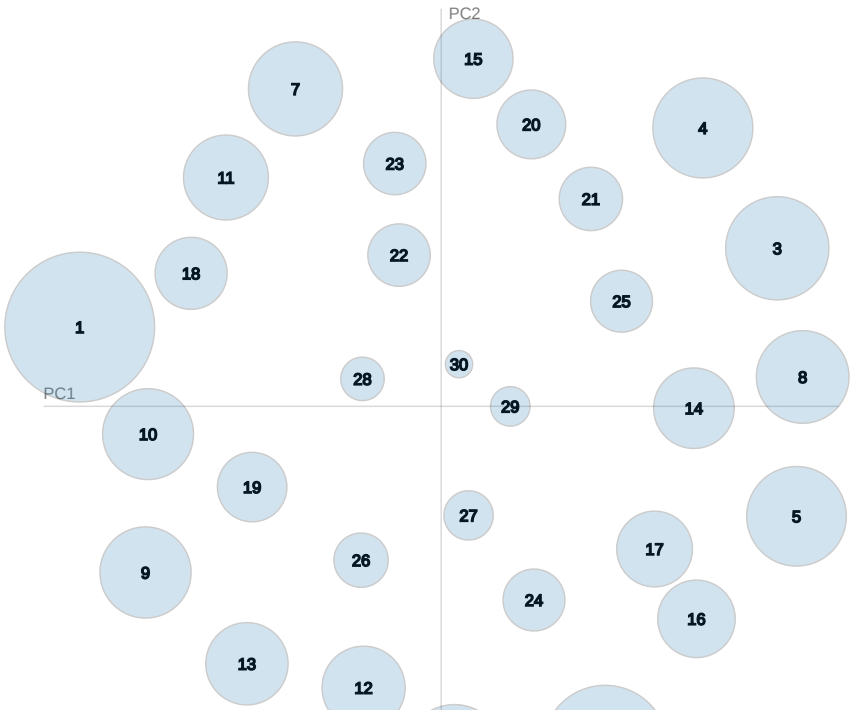
Out[22]:

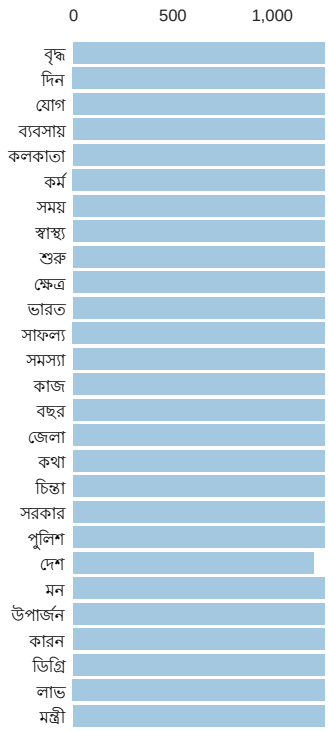
Selected Topic: 0 Previous Topic Next Topic Clear Topic

Slide to adjust relevance metri (2)

λ = 1

Intertopic Distance Map (via multidimensional scaling)





Calcul de spécificité

In [23]:
```
max_specificity, min_specificity = get_higest_lowest_specificity(filtered_bartamanpatrika, 10)
```

In [24]:
```
print('Token les plus spécifiquess:\n')
pprint(max_specificity)
print('\nToken les moins spécifiques:\n')
pprint(min_specificity)
```

```
Token les plus spécifiquess:

{'আসন': 9.387581209964562,
 'কলেজ': 16.87572928751854,
 'চর্চরী': 7.985396726364119,
 'চিত্র': 7.45776581491095,
 'ছাত্র': 16.53630286078982,
 'টাকা': 10.052702666033285,
 'বিভাগ': 6.977808843945238,
 'রাস্তা': 7.5792014157002185,
 'সিনেমা': 14.42726730134741,
 'সীমান্ত': 9.318552883464237}

Token les moins spécifiques:

{'খাবার': -6.1687421867958525,
 'গ্রাহ্য': -5.257513364195549,
 'ফোন': -5.420508507490056,
 'যন্ত্রণা': -6.174903525809364,
 'রোগ': -6.620159514458091,
 'শরীর': -6.140998079367922,
 'শিকার': -7.436707861935691,
 'সোনা': -5.937315752402308,
 'স্বাস্থ্য': -6.842990449898641,
 'হার্ট': -5.629439357111521}
```

Observation

Pour *bartamanpatrika* les termes les plus saillants, représentés par le sujet 1 sont: ***« affaires, succès, avancé, enfant, savoir, réalisation, progressif, religion, réputation, richesse, profession, source, voyage, réunion, défavorable, route, quantité, communication»*** Avec calcul de spécificité, les meilleurs scores sont des termes ***«université»*** et ***«étudiant»***. Nous pouvons interpréter les termes comme représentatifs de l'éducation, la profession, peut-être d'une situation d'une catégorie socio-professionnelle de la société.
Les termes les moins spécifiques, ***nourriture, acceptable, téléphone, douleur, maladie, corps, proie, or, santé, cœur*** parlent plutôt de la santé. Le topic 15 pointe vers des sujets liés au maintien de l'ordre, avec des mots: ***la mort, fermé, le corps, un jeune homme, plainte, infecté, région, peut, difficile, le virus, jeter, panique, brique, inquiétude, la crémation, exclusion***.

Anandabazar

Lecture du corpus et preprocessing

In [25]:
```
anandabazar = "../corpus/txtFiles/anandabazar.txt"
filtered_anandabazar = read_corpus(anandabazar, num_tokens)
filtered_anandabazar = [list for list in filtered_anandabazar if len(list)>0]
```

Génération des bigrammes et trigrammes avec gensim

In [26]:
```
# generation
data_words  = gen_words(filtered_anandabazar)
# print(data_words[9][:200])

# copy_this
bigram_phrases = gensim.models.Phrases(filtered_anandabazar, min_count=5, threshold=50)
trigram_phrases = gensim.models.Phrases(bigram_phrases[filtered_anandabazar], threshold=50)

bigram = gensim.models.phrases.Phraser(bigram_phrases)
trigram = gensim.models.phrases.Phraser(trigram_phrases)

data_bigrams = make_bigrams(filtered_anandabazar)
data_bigrams_trigrams = make_trigrams(data_bigrams)
```

In [27]:
```
print(data_bigrams_trigrams[:3])
```

```
[['রাজ', 'শতাংশ'], ['ইডির', 'দাবি', 'পিতা', 'দায়', 'স্বীকার', 'দায়ী', 'পিতা', 'সংস্থা', 'কর্মী', 'কথা', 'দাবি', 'ইডি'],
['নারী', 'আফতাবে', 'শ্রদ্ধা', 'ঝাল', 'দিল', 'পুলিশ', 'পাতা', 'কথা', 'উল্লেখ']]
```

Filtrage des termes pour conserver uniquement les termes dont les valeurs TF-IDF sont supérieures au seuil spécifié

In [28]:
```
### TF-IDF

id2word = Dictionary(data_bigrams_trigrams)
corpus = [id2word.doc2bow(text) for text in filtered_anandabazar]

word = id2word[[9][:1][0]]
print (word)

corpus=get_corpus(corpus,id2word)
```

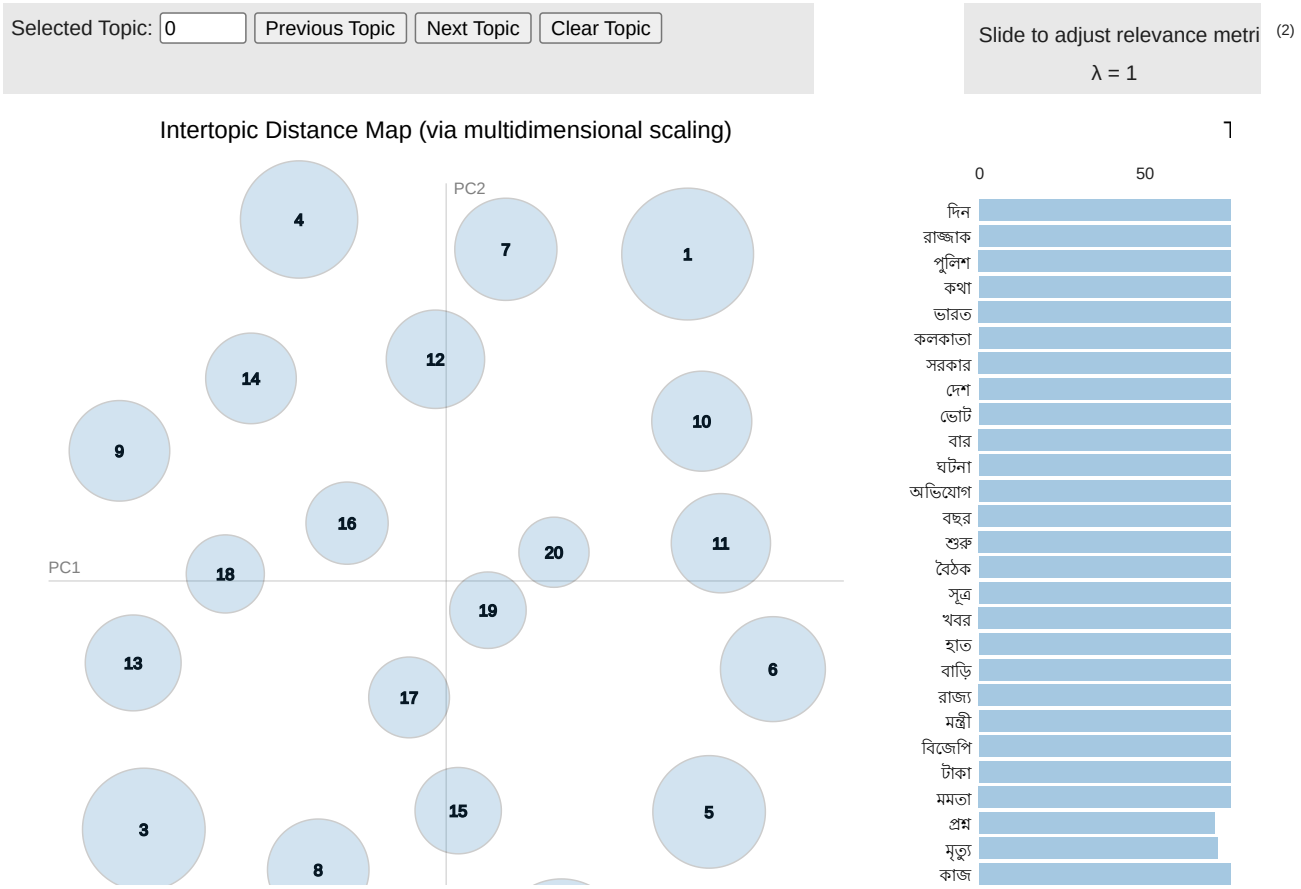
পিতা

Création de modèle LDA et visualisation des *topics*

In [29]:
```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=20,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=200,
                                            passes=10,
                                            alpha="auto")
```

In [30]:
```
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word, mds="mmds", R=30)
vis
```

Out[30]:



In [31]:
```
# pyLDAvis.save_html(vis, '../topic_modeling_anandabazar.html')
```

Calcul de spécificité

In [32]:
```
max_specificity, min_specificity = get_higest_lowest_specificity(filtered_anandabazar, 10)
```

In [33]:
```
print('Token les plus spécifiques:\n')
pprint(max_specificity)
print('\nToken les moins spécifiques:\n')
pprint(min_specificity)
```

```
Token les plus spécifiques:

{'বিধায়': 3.9015847281705778,
 'বিশ্বকোষ': 3.573531194216069,
 'ভূমিকা': 2.7002822513093556,
 'মন্ত্রী': 6.022764907786479,
 'মমতা': 5.655952350898678,
 'মানুষ': 3.3551358922923002,
 'ম্যাচ': 6.0056113508315,
 'রাত': 3.3551358922923002,
 'সংখ্যা': 4.2033701000761505,
 'হার': 4.502508207727971}

Token les moins spécifiques:

{'অভিযোগ': -3.8048958068726706,
 'গুল': -2.5125201232638332,
 'ঘটনা': -2.444196621403663,
 'চাকরি': -2.448597215819409,
 'চিন': -2.5125201232638332,
 'ফোন': -3.055736899466187,
 'বিচার': -2.735923470199232,
 'বিষয়': -2.6846097304884027,
 'শহর': -3.8849559379685097,
 'সরকার': -2.5450620433288833}
```

Observation

Pour le corpus Anandabazar, avec Gensim, nous constatons que le topic 1 (***police, événement, Mamata, nuit, raison, grève, politique, section, ministre, responsabilité, corps, lion, part, centre, prince, vieux***) parle des évènements politiques. Il mentionne la Premier ministre de l'État du Bengale occidental", Mamata. Le sujet 14 pointent également vers la politique et le gouvernement de l'Etat, les termes y figurant sont: ***réunion, congrès, député, district, siège, débat, préparé, illusion, intrusion, entouré, Ram, doute, nez, chinois, Mohan, suspendu, condition, semaine, Murshid, assemblée, combat, étonnement, fumée, but, lieu, cloche, fierté, Mamata***. Le calcul de spécificité semble vérifier cette tendance avec des expressions : ***législation, encyclopédie, rôle, ministre, Mamata, personnes, match, nuit, nombre, taux***. Il parle également de sport. Globalement, ce corpus semble principalement aborder des questions politiques. Les sujets les moins spécifiques ***la politique, la Chine, le gouvernement, le jugement***.

Conclusion

Ce projet a démontre l'efficacité et la puissance des outils TAL existants pour l'extraction et l'automatisation des corpus. Cependant, l'analyse sémantique automatisée s'avère moins satisfaisante que l'analyse manuelle. Nous oberservons d'avantage l'utilisation des termes liés aux ouvriers, agriculteur dans le corpus du *Ganashakti*. Alors que *Anandabazar* semble être un corpus qui traite d'avantage la politique et les affaires de tous les jours. Quant au corpus *Bartamanpatrika*, nous pouvons associer des perception de la société en termes d'accomplissment sociaux : étude, santé, religion progrès, travail.

Malgré ces observations faites à partir des algorithmes, nous ne pouvons être certains de l'exactitude des interprétations faites. Néansmoins, cela marque le début d'un travail intéressant. Les approches utilisées montre certaines limites, dues à la complexité de la tâche et aux moyens informatiques restreints. Plusieurs aspects méritent une attention particulière, et une expérimentation sur les points suivants aurait rendu ce projet plus exhaustif :

- Préciser et élargir la définition du corpus contrastif afin d'englober une gamme plus large de cas d'utilisation et de contextes.
- exploitation des document institutionnels.

- Développer des méthodes plus sophistiquées pour créer des corpus non biaisés, en mettant l'accent sur la neutralité et la représentativité des données.
- Explorer des techniques avancées d'élaboration d'algorithmes pour une extraction plus précise et efficace des caractéristiques du corpus, en tenant compte de la complexité des données et des exigences spécifiques de l'analyse.
- L'expérimentation avec l'inclusion et l'exclusion des entités nommées.

P.S.

- Ce rapport se trouve en version `jupyter notebook` dans le dossier `scripts`, sous le nom `bangla_lda_specificite.ipynb`.
- Il serait recommendable de réduire le nombre de tokens afin de faire un essai «rapide» des traitement.
- Normalement, l'installation des bibliothèques mentionnées dans `requirement.txt` permettent d'exécuter le notebook afin de reproduire les résultats. (Il se peut que quelques bibiliothèques inutiles soient listées dans `requirement.txt`).
- Le package `Textometry` est nécessaire pour les calculs de spécificité, il est sur le git.
- Des remerciements particuliers à Mme. Wang pour son assistance concernant l'encodage et la transformation de police, même si du au manque de temps je n'ai pas pu les appliquer.

Sitographie

- Corpus
- wayback-machine-downloader
- BnLemma
- BNLP
- WayBack Machine
- Medium