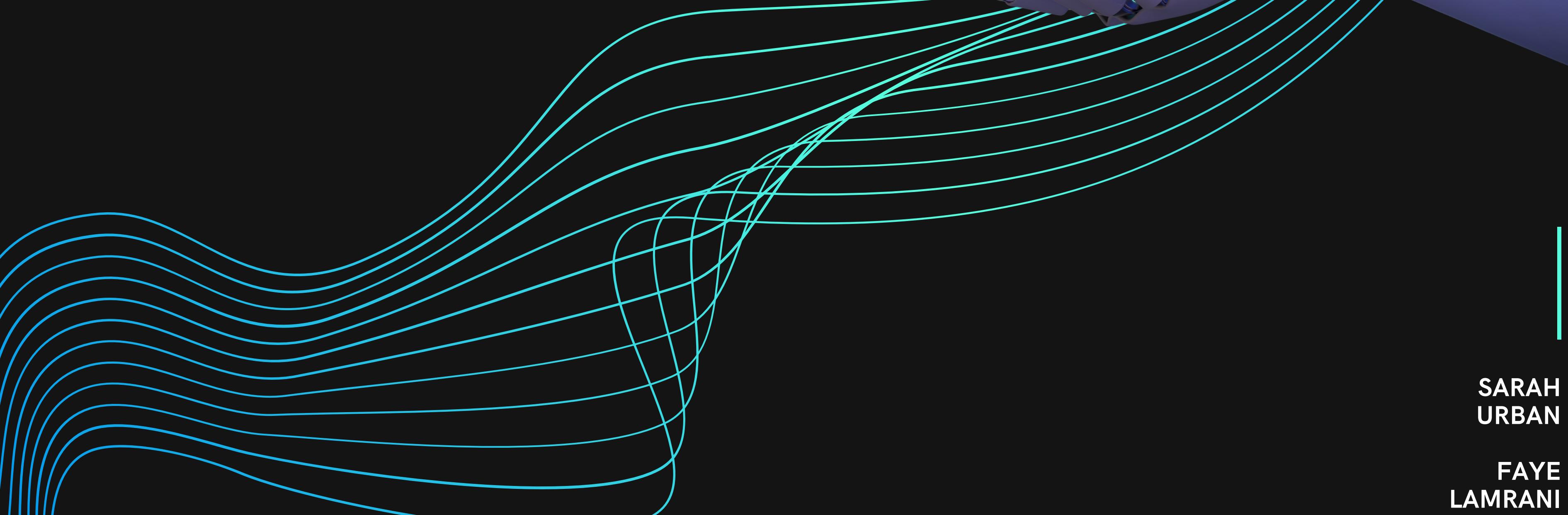
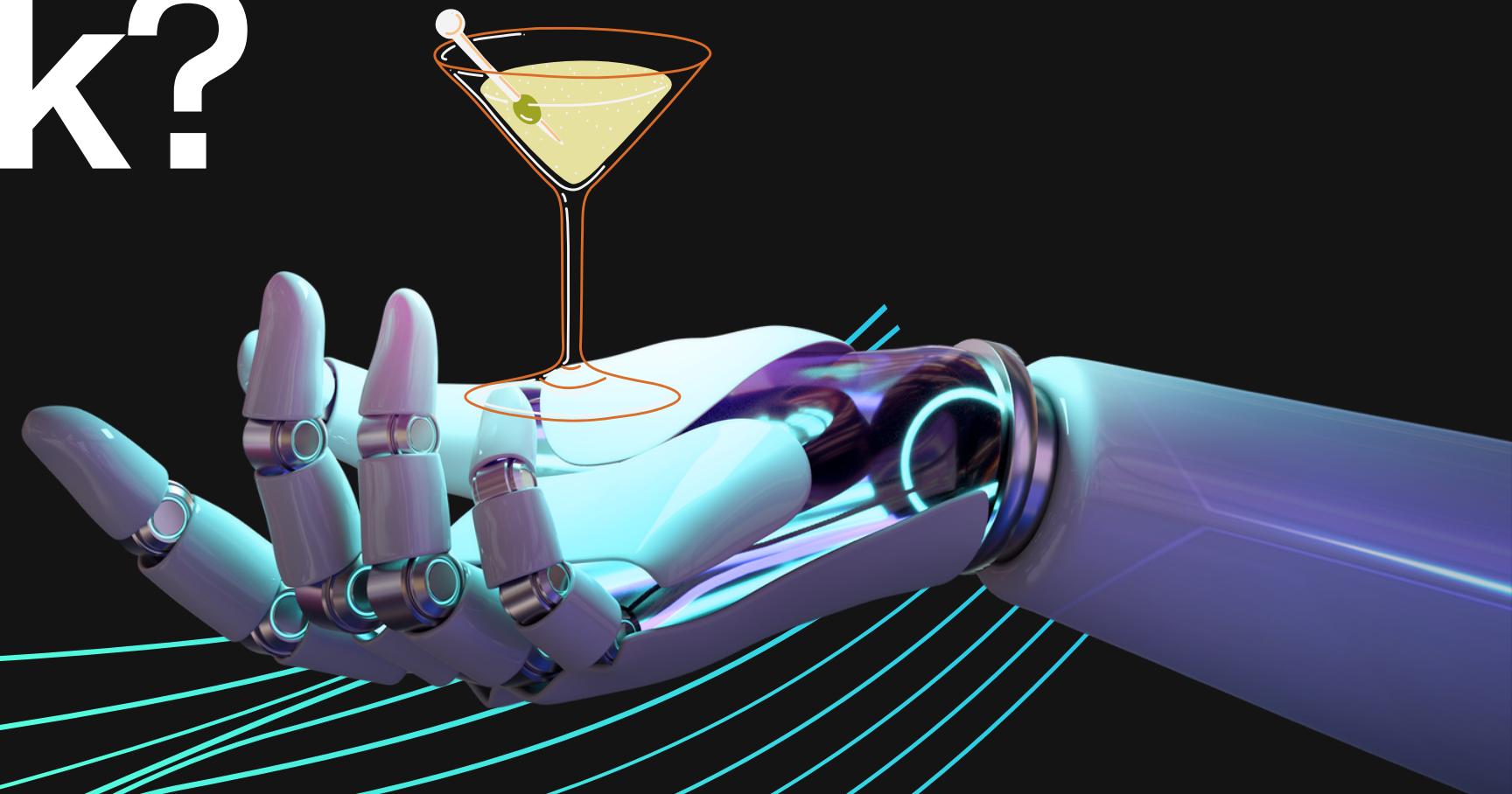


One more drink?

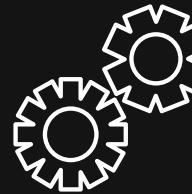
LEGAL DATA ANALYSIS - FINAL PROJECT



SARAH
URBAN

FAYE
LAMRANI

Outline



1

Goals and Approach



2

Phase 1 - Topic Modeling



3

Phase 2 - Throughout the Years



4

Conclusion



Goals and Approach

RESEARCH QUESTION

In what ways are French drinking habits impacted by French Law?

How do French legal databases reflect French drinking habits ?



Goals and Approach



04

PHASE 1 : PICKING A DATASET

- Cour de cassation ?
- Legifrance ?

PHASE 2 : SCRAPPING THE COUR DE CASSATION WEBSITE

- "Inspired" by Professor Charlottin's Github page

PHASE 3 : TOPIC MODELLING AND RESULTS

- Tried (*sometimes unsuccessfully*) to identify most important topics by using different methods

PHASE 4 : SCRAPPING LEGIFRANCE FOR A BROADER OVERVIEW

- Decided to go to Legifrance to include legislations

PHASE 5 : ANALYZING HOW IVRESSE HAS EVOLVED WITH THE LAW

- Plotted the evolution of ivresse in time : in the law, and in the jurisprudence

ALCOHOL IN FRENCH LAW THROUGHOUT THE YEARS

1873

First laws aiming to control drunkenesses

1986

Introduces the possibility of immediate withdrawal of a driver's license

1970

First law against Drinking and Driving

1991

Evin Law



Scrapping the Cour de Cassation website

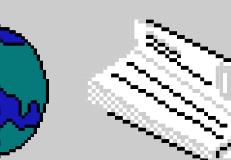
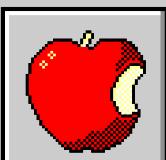
```
#####
#                               SCRAPPING THE COUR DE CASSATION
#####

url = "https://www.courdecassation.fr/recherche-judilibre?search_api_fulltext=ivresse&op=Rechercher"
nlp = spacy.load('fr_core_news_md', disable=["tok2vec", "tagger", "parser", "attribute_ruler", "ner", "textcat"])

main_list = []
dds = [] # a container for the defaultdicts from the decisions
types = [] # keeping track of all categories of text in the database

webpage = requests.get(url) # we connect on the page with docs urls, page by page
soup = BeautifulSoup(webpage.content)

for i in range(79): # we iterate through all the pages
    print(i)
    if i != 0:
        new_url = "https://www.courdecassation.fr/recherche-judilibre?search_api_fulltext=ivresse&op=Rechercher&page=" + str(i)
        webpage = requests.get(new_url)
        soup = BeautifulSoup(webpage.content)
```



[Back to Agenda Page](#)

```

for i in range(79): # we iterate through all the pages
    print(i)
    if i != 0:
        new_url = "https://www.courdecassation.fr/recherche-judilibre?search_api_fulltext=ivresse&op=Rechercher&page=" + str(i)
        webpage = requests.get(new_url)
        soup = BeautifulSoup(webpage.content)

    decisions = soup.find_all("div", class_="decision-item") # And we find the list of docs urls

    for d in decisions:
        # Add link
        lien = d.find("a").get("href")
        sublist = [lien]

        # Add title
        title = d.find("h3").text.split("\n") # We split to obtain the relevant subelements in the title

        formation = d.find("p", class_="decision-item-header--secondary").text.split("-")[0]
        solution = d.find("p", class_="decision-item-header--secondary solution").text

        for x in title + [formation, solution]: # And we add all this to our sublist
            sublist.append(x.strip())

        # Navigate to every decision
        webpage = requests.get("https://www.courdecassation.fr" + lien)

        soup = BeautifulSoup(webpage.content)
        textt = soup.find("div", class_="decision-content decision-content--main")

        text = soup.find("div", class_="decision-content decision-content--main").getText() # To fetch the text of the decision

        people = re.search("AU NOM DU PEUPLE", text) # Cutting if text is too long, and we prefer to cut the entête rather than the text
        sublist.append(text[people.start():6000]) if people is not None else sublist.append(text[:5000]) # A 5000 max length is legit

        main_list.append(sublist) # Adding to main_list, which is thus a list of lists, with each sublist corresponding to a case

        dd = defaultdict(str) # We create a defaultdict to make sure that new entries will be an empty string
        current_type = "" # variable to keep track of the header

        for el in soup.find_all(["h3", "p"]):
            if el.name == "h3" and el.find("button") is not None:
                current_type = el.find("button").getText().strip()
                if current_type not in types:
                    types.append(current_type)
            elif el.name == "p" and "id" in el.attrs: # We make sure this is a "p" element, as some h3 elements without button subels should not be taken into account
                dd[current_type] += el.getText().strip() + "\n----\n"
        dds.append(dd)

```



Scrapping the Cour de Cassation website

```
for el in soup.find_all(["h3", "p"]):
    if el.name == "h3" and el.find("button") is not None:
        current_type = el.find("button").getText().strip()
        if current_type not in types:
            types.append(current_type)
    elif el.name == "p" and "id" in el.attrs: # We make sure this is a "p" element, as some h3 elements without button subels should not be taken into account
        dd[current_type] += el.getText().strip() + "\n-----\n"
dd.append(dd)

for e, d in enumerate(dd):
    for tt in types:
        data = d[tt]
        main_list[e].append(data[:5000])

cassdf = pd.DataFrame(main_list, columns=["URL", "Date", "Cour", "ID", "Formation", "Solution", "Text"] + types)
```

Topic Modeling

Goal : Identifying whether certain topics are particularly present in French breaches to the law in terms of drinking

4 Main Steps :

- Process the text with Spacy
- Word Cloud
- Latent Dirichlet Allocation (LDA)
- BERT



Topic Modeling

- Process the text with Spacy

```
#=====
#                      TOPIC MODELLING
#=====

# STEP 1 : process text using spacy

def spacy_process(text): # We first prepare the text by using spacy's token elements to remove stop words and punctuation
    doc = nlp(re.sub(r"\(\)", " ", text)) # We transform the text with spacy

    filtered_sentence = [] # Empty list for the tokens we'll want to keep

    punctuations = [ "?", ":" , "!" , ".", "," , ";" , "-" , "(" , ")" , "[" , "]" ] # A list of punctuation
    banned_words = [ "publique" , "palais" , "justice" , "base" , "légale" , "état" , "ivresse" , "alcoolique" , "avocat" , "général" , "peuple" , "français" , "débats" , "article" , "audience" , "chambre" , "conseiller" , "cour" , "président" , "cour cassation" , "arrêt" , "code" , "procédure" , "juridiction" , "moyen" , "pourvoi" , "cassation" , "appel" , "arrêt" ]

    for token in doc:

        if token.is_stop is False and token.lemma_ not in punctuations and token.is_alpha and token.text.lower() not in banned_words:
            filtered_sentence.append(token.lemma_)

    return " ".join(filtered_sentence)

# Apply the function to the text and store in newly created column CText
cassdf["CText"] = cassdf.Text.apply(spacy_process)
```

Banned Words

```
banned_words = [ "publique" ,
"palais" , "justice" , "base" , "légale" ,
"état" , "ivresse" ,
"alcoolique" , "avocat" , "général" ,
"peuple" , "français" , "débats" ,
"article" , "audience" , "chambre" ,
"conseiller" , "cour" , "président" ,
"cour cassation" , "arrêt" , "code" ,
"procédure" , "juridiction" , "moyen" ,
"pourvoi" , "cassation" , "appel" ,
"arrêt" ]
```

Topic Modeling

- Word Cloud



```
# STEP 2 : visualize term frequency with word cloud

long_string = ','.join(list(cassdf["CText"].values))
wordcloud = WordCloud(background_color="white", max_words=5000, contour_width=3, contour_color='blue').generate(long_string)

plt.imshow(wordcloud, interpolation = "bilinear")
plt.axis("off")
plt.show()
```

WordClouds...



garde vue victime tenue paris français criminelle
rendu rapport conduite empire motifs manque xpart
motif accident employeur conduite véhicule
déclaration reporter griffon prévenu mois emprisonnement
faits criminelle jour été statuant formé
suspension permis condition absence
permis conduire francs mandats correctionnelle date
non texte décision motivation mémoire nom français
dernier z pris violation civile professionnelle
société civile conducteur défaut motif
salarié criminelle tenue partie civile forme
conversion europeenne rapport motif
griffon motif
réfugié réfugié
émission émission avec objet refus consentir
rapport motif
griffon motif
réfugié réfugié
émission émission avec objet refus consentir

Topic Modeling

```
# STEP 3 : topic modeling with LDA. Source : https://medium.com/analytics-vidhya/modeling-with-latent-dirichlet-allocation-3b198f1a7bae
def lda_topic_model(cassdf):

    count = CountVectorizer(max_df=.1, max_features=5000)
    X = count.fit_transform(cassdf[\"CText\"].values)

    lda = LatentDirichletAllocation(n_components=10, random_state=123, learning_method= "batch")
    X_topics = lda.fit_transform(X)

    n_top_words = 5
    feature_names = count.get_feature_names()

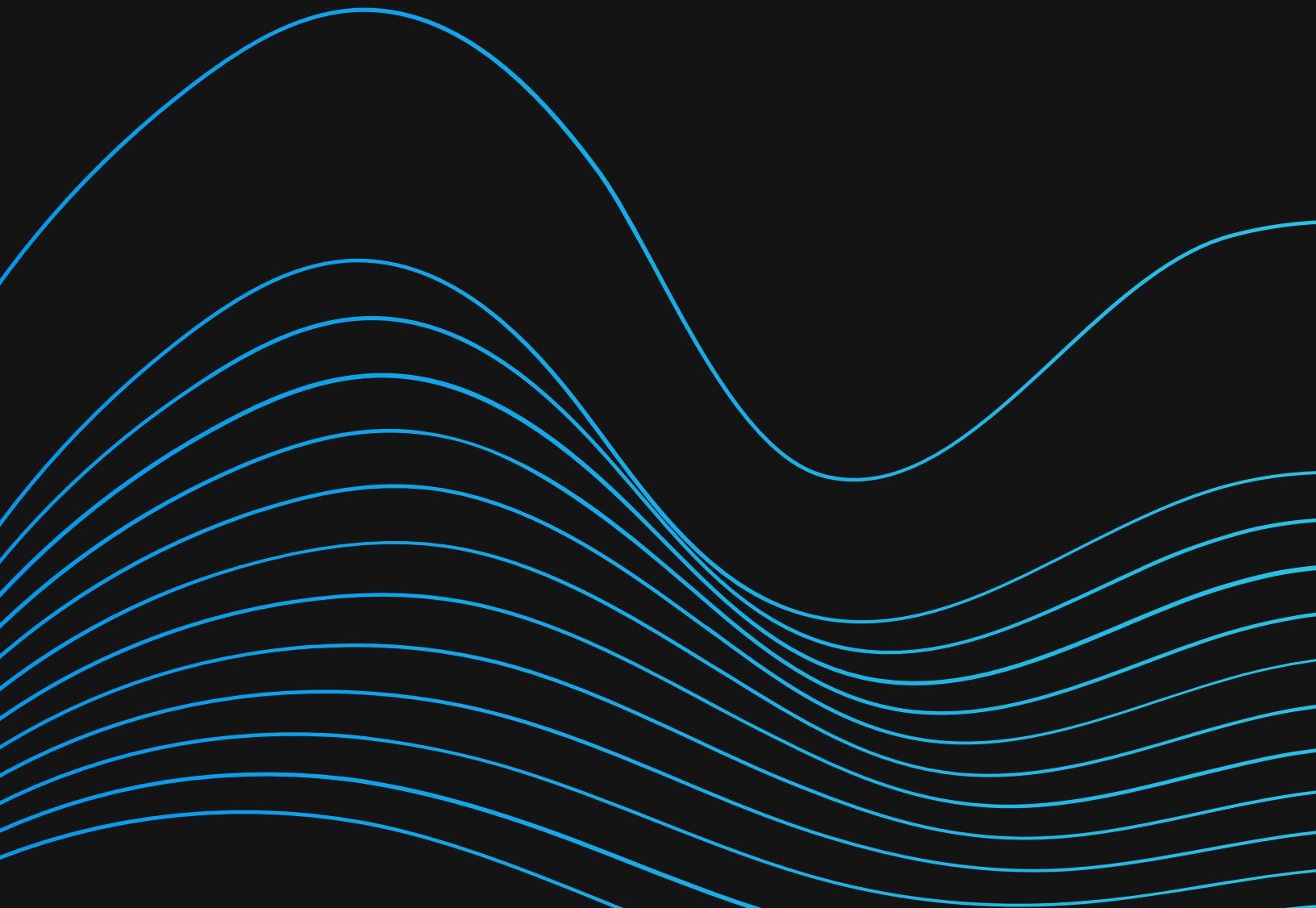
    for topic_idx, topic in enumerate(lda.components_):
        print("Topic %d:" % (topic_idx + 1))
        print(" ".join([feature_names[i]
                      for i in topic.argsort()
                      [-n_top_words - 1:-1]]))

lda_topic_model(cassdf) # call the lda function
```



- Latent Dirichlet Allocation (LDA)

LDA results



- Topic 1: assuré fausse risque intentionnelle axa
- Topic 2: notification garde épreuve autorité air
- Topic 3: accusation coups instruction volontaires assises
- Topic 4: épouse divorce mari époux monsieur
- Topic 5: garde débits audition policiers interpellation
- Topic 6: caisse demeurant employeur grave primaire
- Topic 7: clause exclusion assuré décès existe
- Topic 8: analyse gendarmes maîtrise prélèvement chaussée
- Topic 9: salarié licenciement employeur entreprise règlement
- Topic 10: procedure defaut prevenu vehicule penale

Topic Modeling

- BERT

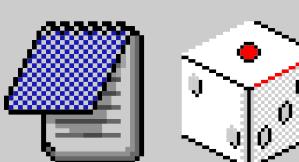
```
# STEP 4 : topic modeling with BERT. Source : class GitHub
def bert_topic_model(cassdf):

    topic_model = BERTopic(embedding_model=nlp, n_gram_range=(1, 3), min_topic_size=3, nr_topics="auto") # We set up the topic model
    time.sleep(4)
    topics, probs = topic_model.fit_transform(cassdf.CText.values.tolist())

    topic_model.get_topic_info() # We visualise the top terms for each topics
    cassdf["Topic"] = topics
    cassdf.to_clipboard(index=False, encoding="utf8")

    fig = topic_model.visualize_topics()
    pio.show(fig)

bert_topic_model(cassdf) # call the bert function
```



[Back to Agenda Page](#)

BERT results (the most relevant ones)

Topic 1

Words: état | véhicule | conduite | alcoolique | mois

Size: 110

Topic 2

Words: contrat | travail | société | salarié | licenciement

Size: 106

Topic 5

Words: arret | responsabilite | faute | accident | victime

Size: 18

Topic 6

Words: violences | coups | été | faits | victime

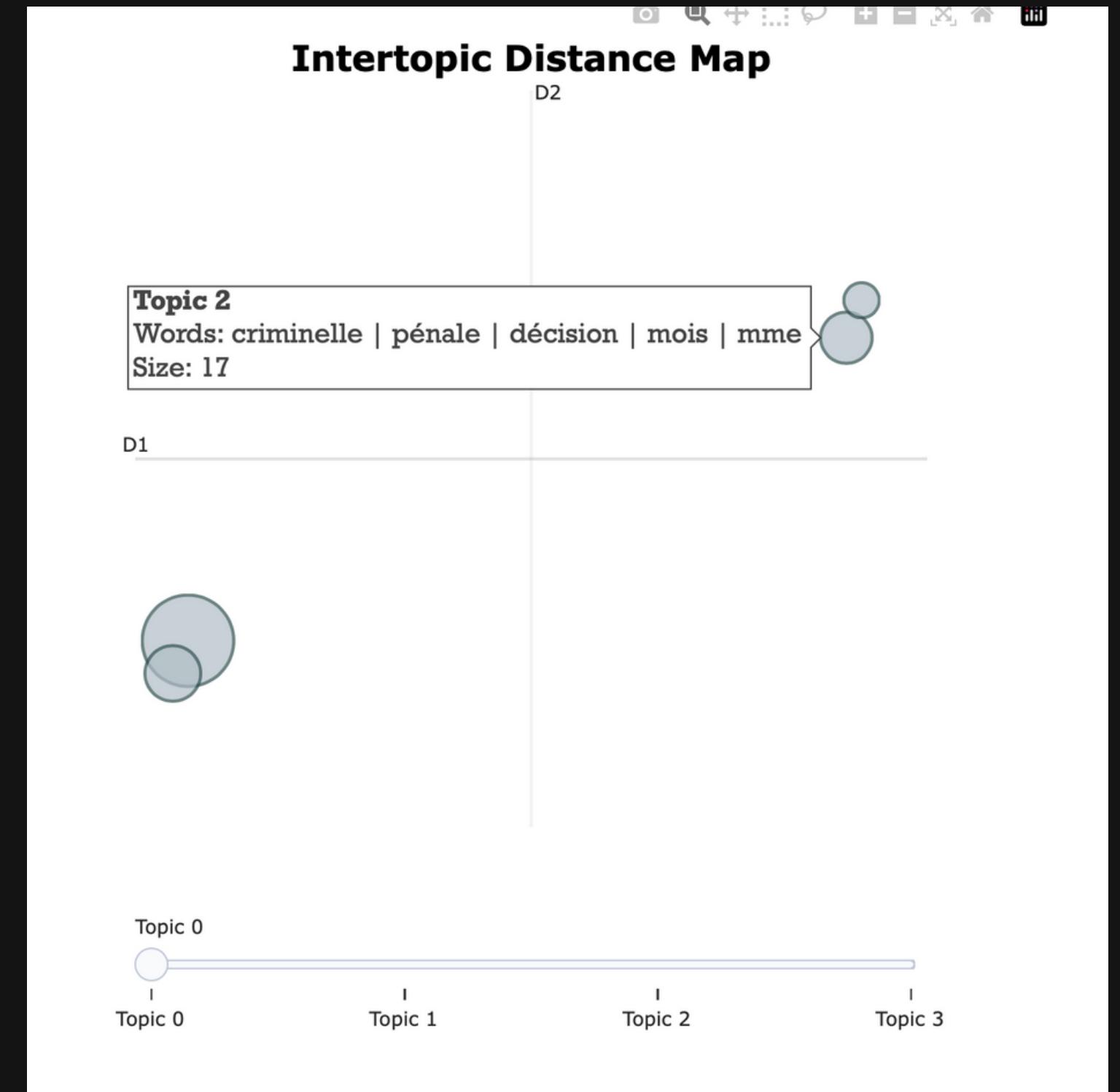
Size: 18

The proces

Last results we got were less relevant...

Overall topic modeling results :

- drinking and driving...
- domestic violence
- workplace issues



PART 2 Trying to establish a link between drinking laws and jurisprudence with Legifrance

Légifrance

Le service public de la diffusion du droit

Informations de mises à jour | Gestion des cookies | Nous contacter | Activer l'aide sur la page

DROIT NATIONAL EN VIGUEUR | PUBLICATIONS OFFICIELLES | AUTOUR DE LA LOI | Droit et jurisprudence de l'Union européenne | Droit international

Effectuer une recherche dans :

Tous les contenus | Dans tous les champs | Ex. : L. 121-1, CGI, 10-15056, dol, majeurs protégés | Q | RECHERCHE AVANÇÉE

VOTRE RECHERCHE

TOUS LES CONTENUS

Dans tous les champs | ivresse | Relancer la recherche | Q

AFFINER LA RECHERCHE

Par fonds

1586 résultat(s) trouvé(s) au 12/12/2022 | Afficher 10 résultats par page |

Jurisprudence constitutionnelle

Décision 2012-253 QPC - 08 juin 2012 - M. Mickaël D. [ivresse publique] - Conformité - réserve

[...] Considérant qu'aux termes de l'article L. 3341-1 du code de la santé publique : « Une personne trouvée en état d'ivresse dans les lieux publics est, par mesure de police, conduite à ses frais dans le local [...] Considérant que, selon le requérant, en permettant que les personnes trouvées sur la voie publique en état d'ivresse puissent être privées de leur liberté pour une durée indéterminée par une mesure de [...]

Jurisprudence judiciaire

Cour de cassation, criminelle, Chambre criminelle, 21 juin 2017, 16-84.158, Publié au bulletin

[...] X..., dans la chambre occupée par ce dernier ; que les services de police, immédiatement alertés, ont procédé à l'arrestation de ce client, qui était en état d'ivresse, l'ont conduit au commissariat et [...] entaché sa décision d'une



Scrapping Legifrance

- setting up the scrapping : links and empty lists



```
data_scraped = []

lien_ivresse = "https://www.legifrance.gouv.fr/search/all?tab_selection=all&searchField=ALL&query=ivresse&page=1&init=true"
lien_part1 = "https://www.legifrance.gouv.fr/search/all?tab_selection=all&searchField=ALL&query=ivresse&searchProximity=&searchType=ALL&isAdvancedResult=&isAdvancedRe
lien_part2 = "&tab_selection=all"

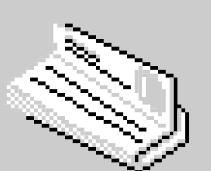
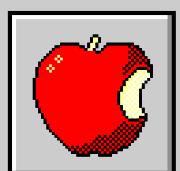
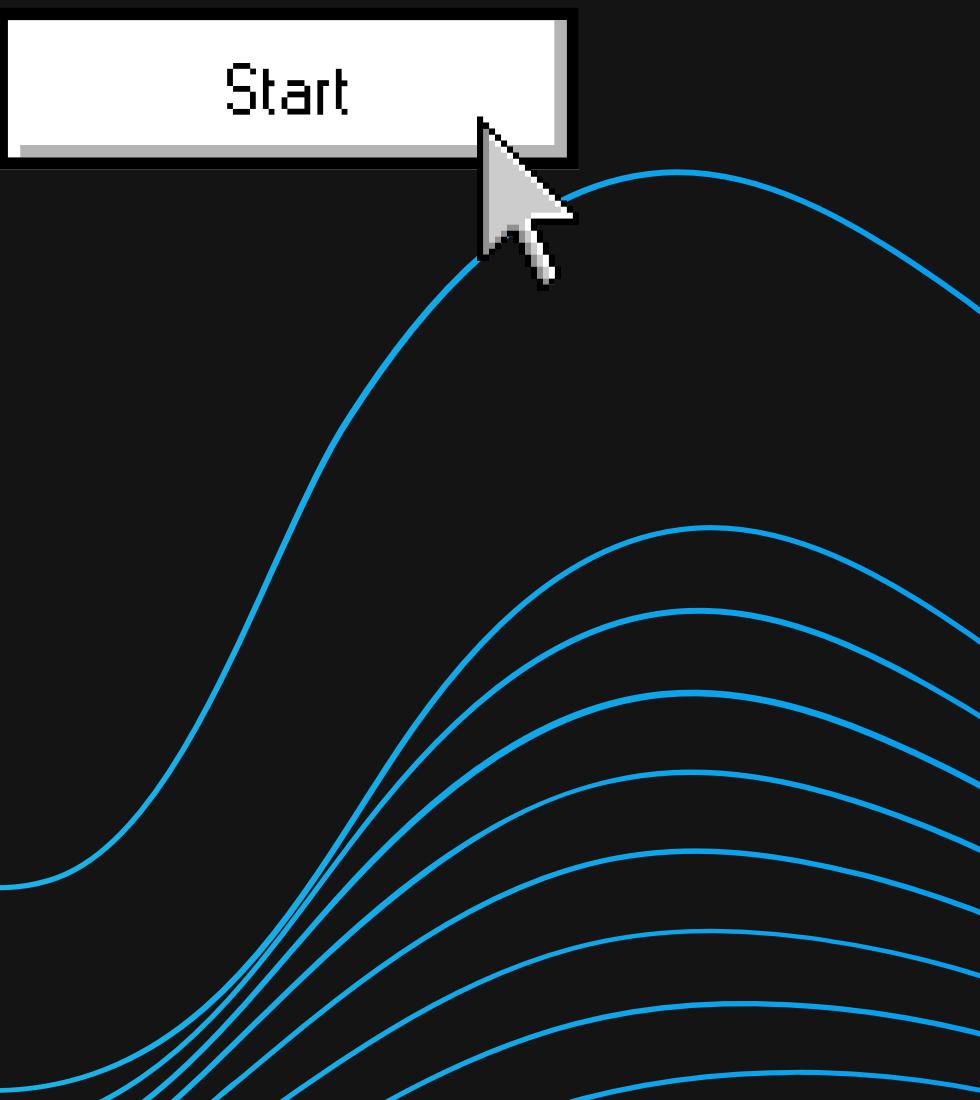
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

entry_list = []
list_of_lists = []
base_link = "https://www.legifrance.gouv.fr"
```

```

for i in range(1,160):
    print(i)
    lien = lien_part1 + str(i) + lien_part2
    time.sleep(1)
    driver.get(lien_part1 + str(i) + lien_part2) # Get the specific law code we are interested, using the dict defined above
    time.sleep(1)
    soup = BeautifulSoup(driver.page_source, features = "lxml") # Read HTML, pass it to a soup object
    results= soup.find_all(class_= "result-item")
    for result in results: # Iterate over articles one by one
        # EXTRACT LINK AND GET TEXT ON PAGE
        if(result.find("a")["href"]):
            entity_link = base_link + result.find("a")["href"]
            driver.get(entity_link)
            """
            try:
                our_text = driver.find_elements(By.CLASS_NAME, 'content-page').text
            except:
                try:
                    our_text = driver.find_elements(By.XPATH, '//*[@id="main"]/div/div/div[2]/div[4]/div').text
                except:
                    our_text = "No text found"
            else:
                entity_link ="NA"
            # EXTRACT TITLE
            if len(result.find_all("a", class_ = "name-result-item")) != 0:
                title = result.find_all("a", class_ = "name-result-item")[0].text
            else:
                title = "NA"
            # EXTRACT CATEGORY
            if result.find_all("a"):
                catégorie = result.find_all("a")[0].text
            # EXTRACT DATE : either in title or in category
            if re.search("\d{2}/\d{2}/\d{4}",title):
                date = re.findall("\d{2}/\d{2}/\d{4}",title)[0]
            elif re.search("\d{1,2}\s[a-zA-Z]{3,9}\s\d{4}",title):
                date = re.findall("\d{1,2}\s[a-zA-Z]{3,9}\s\d{4}", title)[0]
            elif re.search("\d{1,2}\s[a-zA-Z]{3,9}\s\d{4}",catégorie):
                date = re.findall("\d{1,2}\s[a-zA-Z]{3,9}\s\d{4}",catégorie)[0]
            else:
                date = "NA"
            # CREATE LIST WITH EXTRACTED DATA
            our_list = [catégorie, entity_link, title, date] # our text removed at end
            list_of_lists.append(our_list)
    driver.close()
df = pd.DataFrame(list_of_lists, columns=["Category", "Link", "Title", "Date"]) # text removed at end
df.to_csv("ivresse.csv", encoding="utf8")

```



Scraping Legifrance

- trying to extract information from Legifrance : a failed attempt at extracting text from every page

```
for i in range(1,160):  
    print(i)  
    lien = lien_part1 + str(i) + lien_part2  
    time.sleep(1)  
    driver.get(lien_part1 + str(i) + lien_part2) # Get the specific law code we are interested, using the dict defined above  
    time.sleep(1)  
    soup = BeautifulSoup(driver.page_source, features = "lxml") # Read HTML, pass it to a soup object  
    results= soup.find_all(class_ = "result-item")  
    for result in results: # Iterate over articles one by one  
        # EXTRACT LINK AND GET TEXT ON PAGE  
        if(result.find("a")["href"]):  
            entity_link = base_link + result.find("a")["href"]  
            driver.get(entity_link)  
            """  
            try:  
                our_text = driver.find_elements(By.CLASS_NAME, 'content-page').text  
            except:  
                try:  
                    our_text = driver.find_elements(By.XPATH, '//*[@id="main"]/div/div/div[2]/div[4]/div').text  
                except:  
                    our_text = "No text found"""  
        else:  
            entity_link ="NA"
```

Scrapping Legifrance

- finding information part 2 and putting everything into a dataframe

```
# EXTRACT CATEGORY
if result.find_all("a"):
    catégorie = result.find_all("a")[0].text

# EXTRACT DATE : either in title or in category
if re.search("\d{2}/\d{2}/\d{4}",title):
    date = re.findall("\d{2}/\d{2}/\d{4}",title)[0]

elif re.search("\d{1,2}\s[a-zA-Z]{3,9}\s\d{4}",title):
    date = re.findall("\d{1,2}\s[a-zA-Z]{3,9}\s\d{4}", title)[0]

elif re.search("\d{1,2}\s[a-zA-Z]{3,9}\s\d{4}",catégorie):
    date = re.findall("\d{1,2}\s[a-zA-Z]{3,9}\s\d{4}",catégorie)[0]
else:
    date = "NA"

# CREATE LIST WITH EXTRACTED DATA
our_list = [catégorie, entity_link, title, date] # our text removed at end
list_of_lists.append(our_list)

driver.close()
df = pd.DataFrame(list_of_lists, columns=["Category", "Link", "Title", "Date"]) # text removed at end
df.to_csv("ivresse.csv", encoding="utf8")
```

Some necessary Data Cleaning

```
#=====
# DATA CLEANING
#=====

df = pd.read_csv("ivresse.csv", encoding = "utf8", header = "infer")
print(df.head())

# We also delete the empty columns.
df = df.drop(["Unnamed: 0"], axis=1)

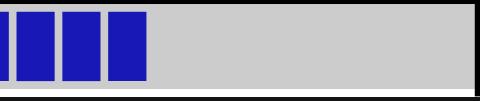
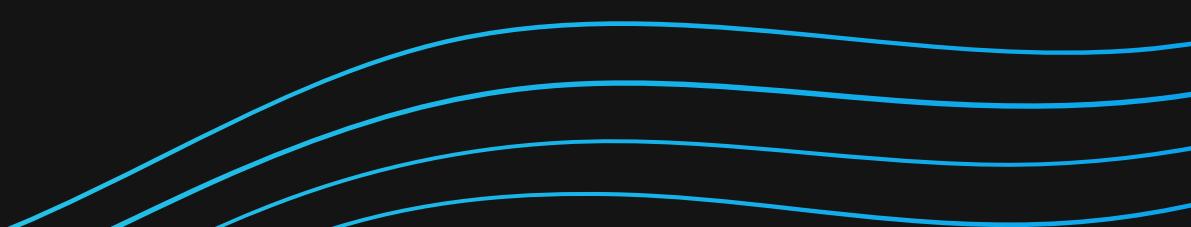
df = df.dropna(how = 'all')

# TRANSFORM DATES INTO DATE TIME OBJECTS !!
format_data = "%d/%m/%Y"
for index, el in df["Date"].items():
    if type(el) == str:
        if re.search("\d{1,2}/\d{2}/\d{4}",el):
            df["Date"][index] = datetime.strptime(el, format_data).date()

    else:
        df["Date"][index] = dateparser.parse(el).date()
        #el = datetime.strptime(el, format_data)

df["Year"] = 0 # default number
# Add a column with year :
for index, el in df["Date"].items():
    if(type(el) != float):
        df["Year"][index] = el.year
```

```
#=====
#PART 2 = TIME ANALYSIS -- THE EVOLUTION
#=====

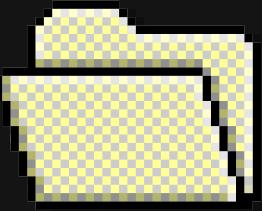
# create larger categories |  
```

Time analysis

- Extracting relevant information
- Split into jurisprudence and laws

Filter Then plot

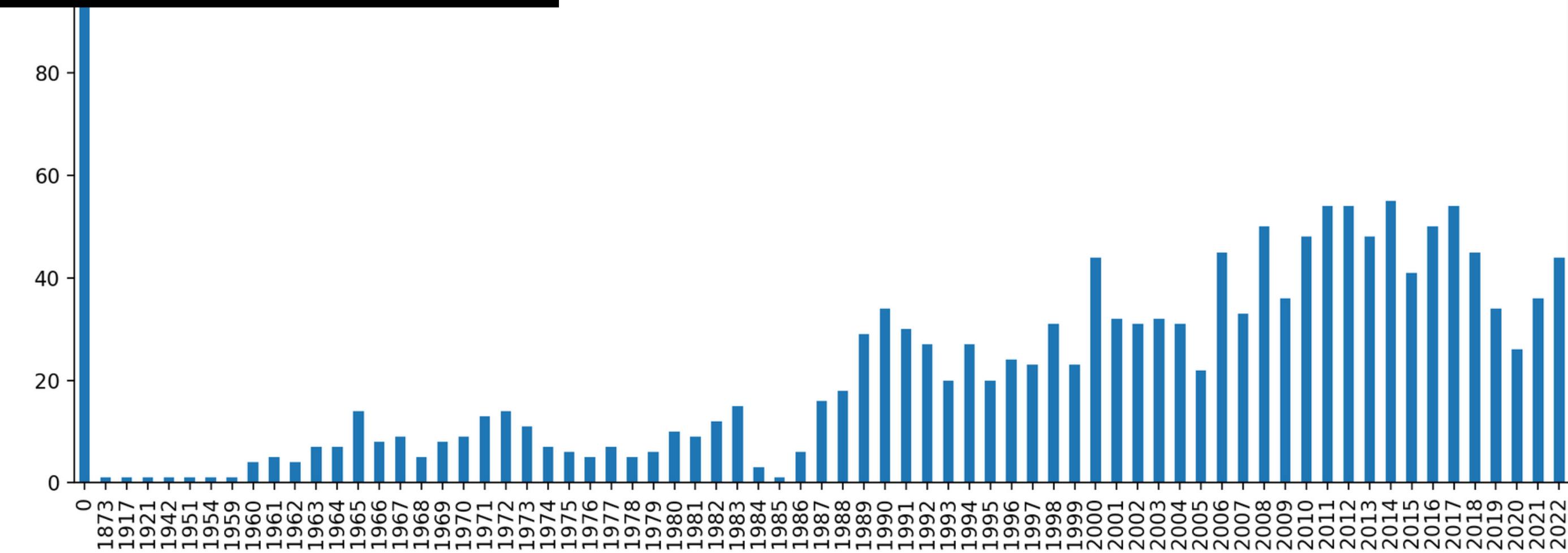
```
# Filter by jurisprudence or loi :  
  
jp_filter = (new_df["Category"] == "Jurisprudence")  
jp_df = new_df[jp_filter]  
  
laws_filter = (new_df["Category"] == "Laws")  
laws_df = new_df[laws_filter]  
  
jp_df.groupby("Year").size().plot(kind = 'bar')  
plt.show()  
  
laws_df.groupby("Year").size().plot(kind = 'bar')  
plt.show()
```



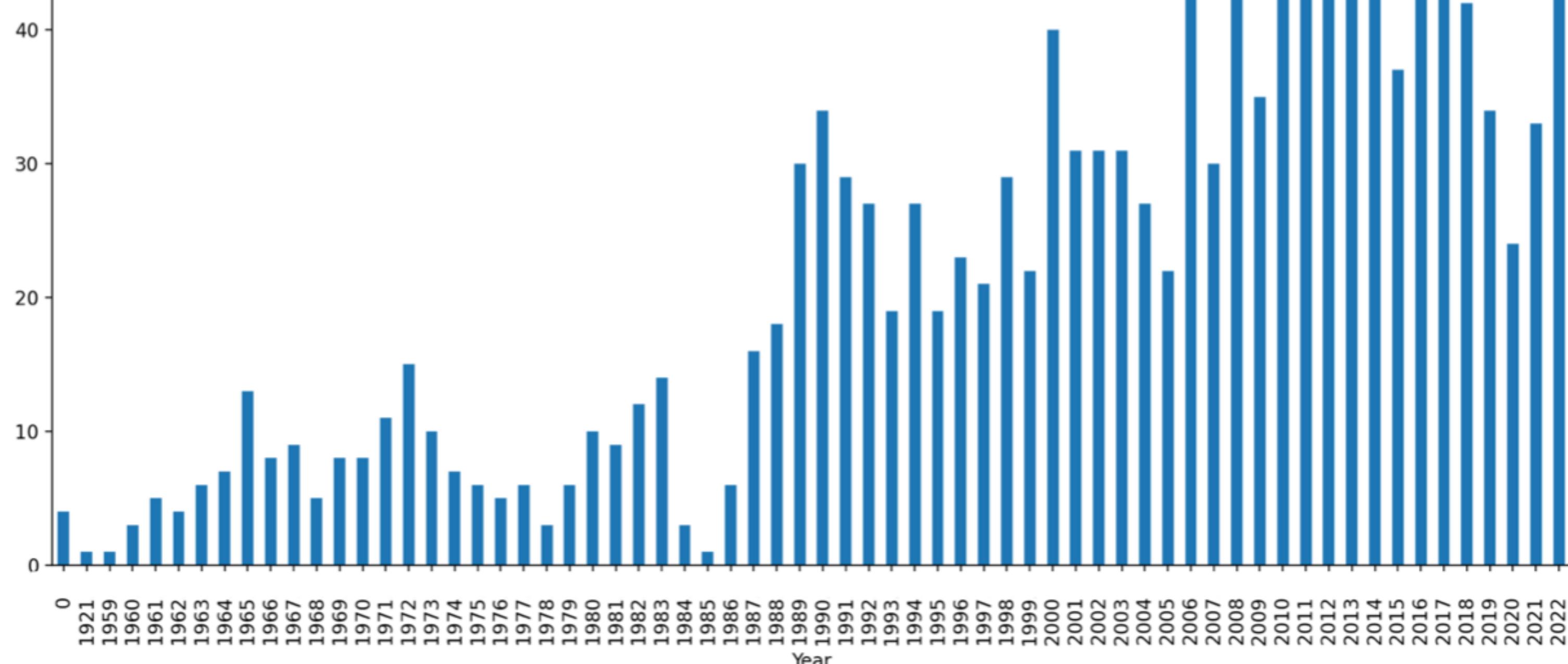
Area for improvement:
tried to plot all on the same graph, but it was complicated because there are
way less laws than jurisprudence

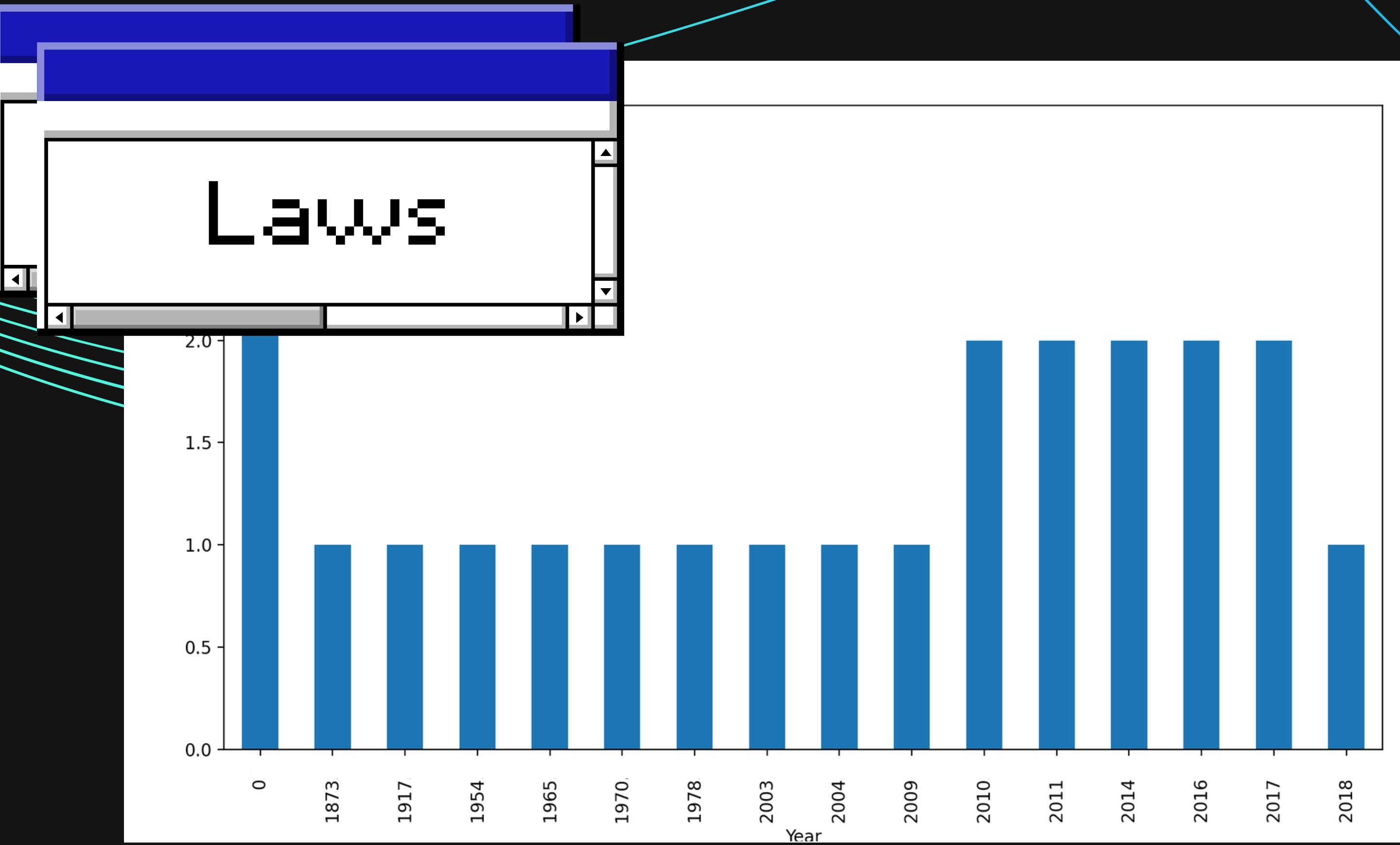
Results with all the data from Legifrance

previous results



Jurisprudence





1873

First laws aiming to control
drunkenness

1986

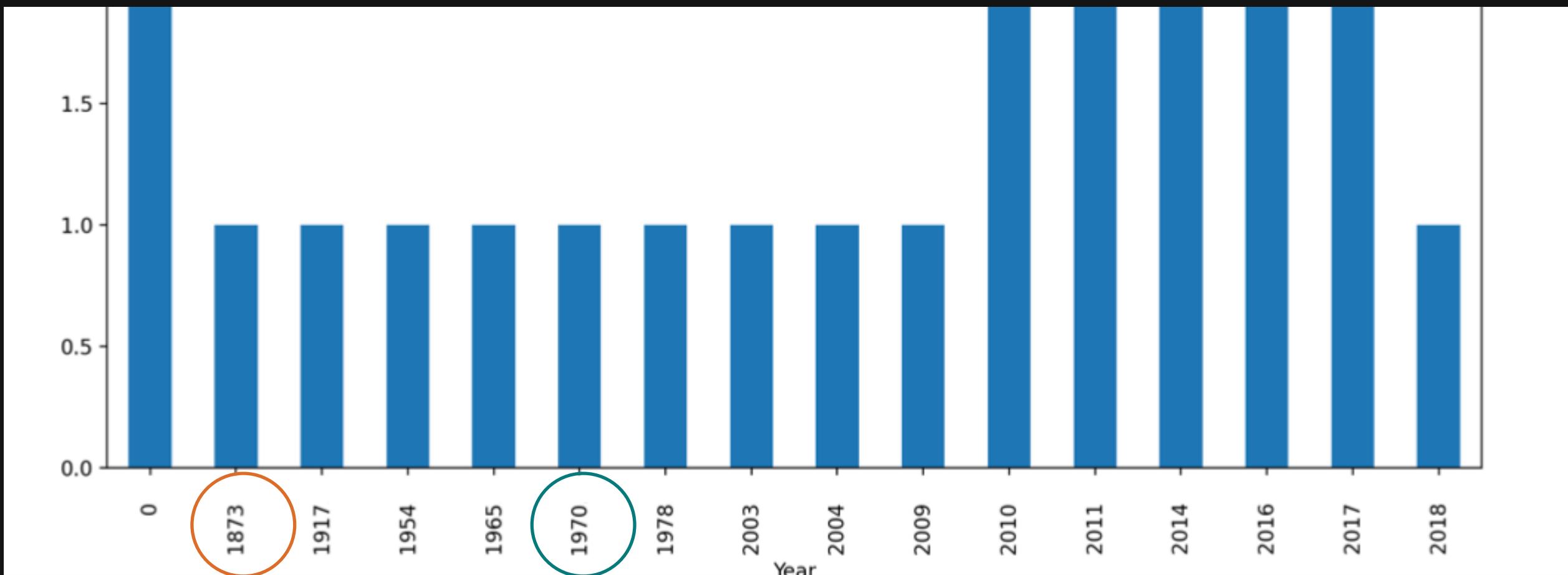
Introduces the possibility
of immediate withdrawal of
a driver's license

1970

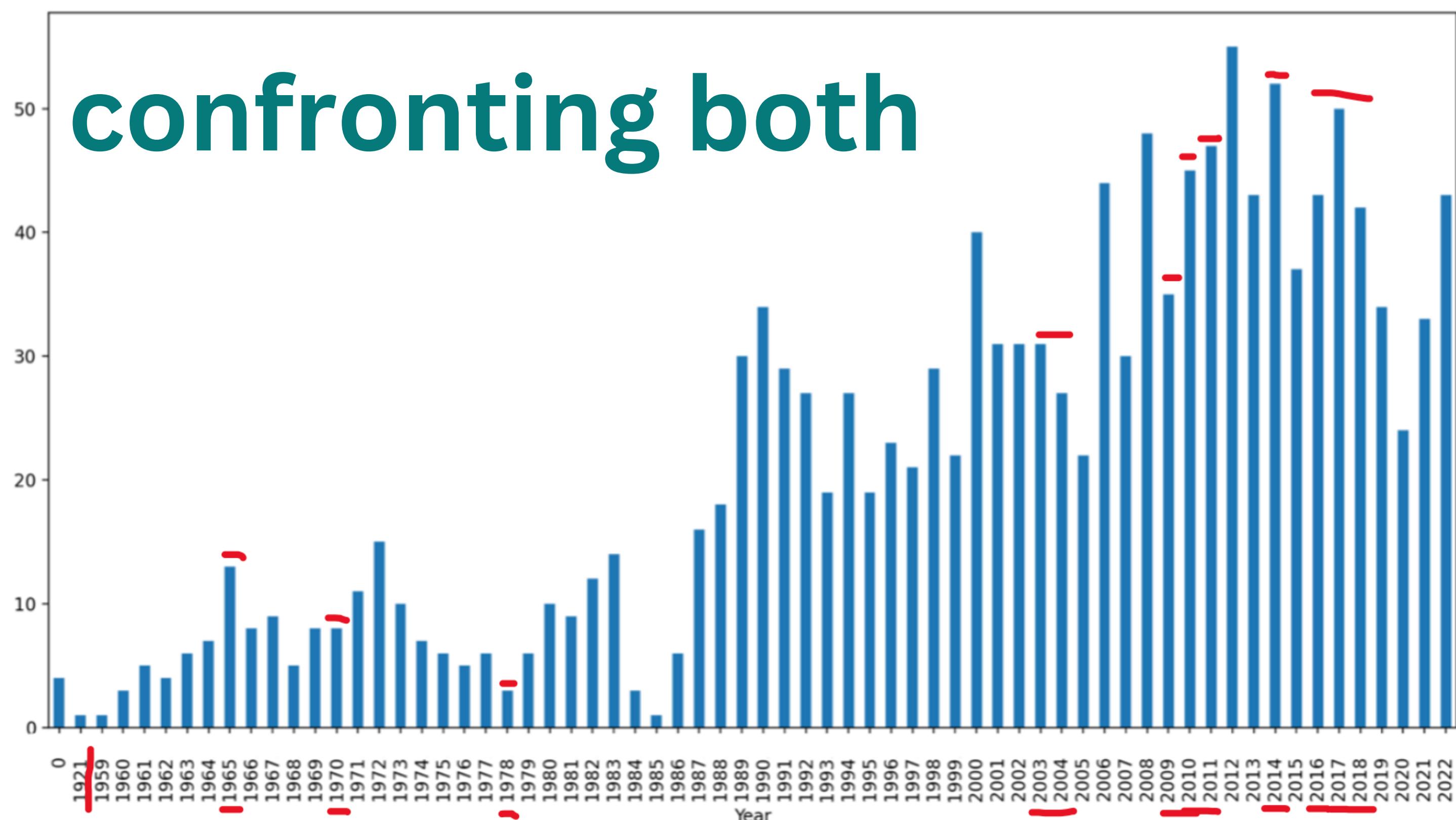
First law against Drinking
and Driving

1991

Evin Law



confronting both



Conclusion

- some interesting results with topic modeling : drunk driving was unfortunately unsurprising...
- a time evolution algorithm that needs fine tuning... but some findings as well

Though our results fluctuated a lot, the numbers were always rising : result of more laws or more debauchery ?

