Intro

This project is made by Open University Computer Science students Vitaly Chait and Gabriel Shapiro. We debated on what our project should be about and we landed on the idea of profanity identification within text. We both saw the need for this type of technology to be implemented within the social media universe and we were both very interested to get to work.

The Problem / Solution

There is a constant rise to the number of devices connected to the web (IOT) and the content being spread by different people across the globe. Also, the starting age of the user is constantly decreasing to our new reality, a reality where every elementary school and even kindergarten kids are already surfing in the open web alone without any parent supervising their activity.

This leads us to the idea of our final data science & machine learning project.

The project will create a scoring system, that will give a pass \ no pass to content loading to its interface. Underage browsers will be able to see content that is suitable for their age by only including text that matches their threshold of profanity. There are many obstacles that must be tested for in order for the model to have a good reliability rating of correct classification. The english language is very safisticated with its grammar, and the meaning of a sentence can change with one word or one symbol. We will be working with datasets of collected sentences from the internet that we will be able to register inside our testing model and classify each sentence with a profanity grade.

INIT and prerequisites

Download and install Visual Studio (https://visualstudio.microsoft.com/downloads/)

If you have an Nvidia GPU you are welcomed to download CUDA

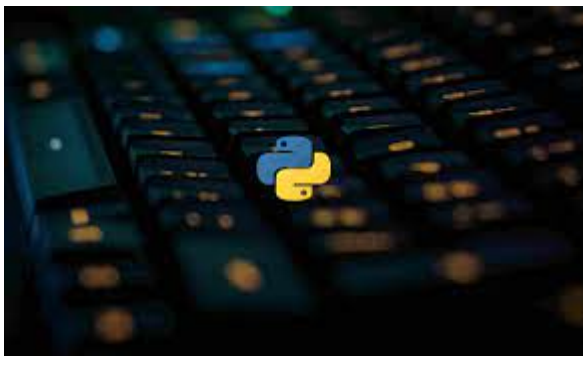
CUDA (https://developer.nvidia.com/cuda-downloads) + cudnn (https://developer.nvidia.com/cudnn)

Libraries

In [1]:
```
import platform
import os

try:
    os.add_dll_directory("C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v11.2/bin")
except:
    print("")

print("Python version: {:>22}".format(platform.python_version()))
print(os.getcwd())

try:
    import scrapy
except:
    !pip3 install scrapy
print("scrapy version: ", scrapy.__version__)

try:
    import cv2
except:
    !pip3 install cv2
print("OpenCV version: ",cv2.__version__)

try:
    import pandas
except:
    !pip3 install pandas
print("pandas version: ", pandas.__version__)

try:
    import PIL
except:
    !pip3 install pillow
print("PIL version: ", PIL.__version__)

try:
    import numpy
except:
    !pip3 install numpy
print("numpy version: ", numpy.__version__)

try:
    import joblib
except:
    !pip3 install joblib
print("joblib version: ", joblib.__version__)

try:
    import sklearn
except:
    !pip3 install sklearn
print("sklearn version: ", sklearn.__version__)

try:
    import tensorflow
except:
    !pip3 install tensorflow
print("tensorflow version: ", tensorflow.__version__)

try:
    import matplotlib
except:
    !pip3 install matplotlib
print("matplotlib version: ", matplotlib.__version__)

try:
    import IPython
except:
    !pip3 install IPython
print("IPython version: ", IPython.__version__)

try:
    import spacy
except:
    !pip3 install spacy
print("spacy version: ", IPython.__version__)

try:
    import seaborn
except:
    !pip3 install seaborn
print("seaborn version: ", seaborn.__version__)


%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

```
from IPython.display import display
```

```
Python version:  3.9.7
C:\Users\vital\Desktop\uni\data_science\Untitled Folder
scrapy version:  2.5.0
OpenCV version:  4.5.1
pandas version:  1.2.4
PIL version:  8.2.0
numpy version:  1.19.5
joblib version:  1.0.1
sklearn version:  0.24.2
tensorflow version:  2.6.0
matplotlib version:  3.4.3
IPython version:  7.27.0
spacy version:  7.27.0
seaborn version:  0.11.2
```



GPU_ENABLED - set True or False the parameter below for activation

In [2]:
```
GPU_ENABLED = True # Change to False if you don't have GPU
```

In [3]:
```
if GPU_ENABLED:
    physical_devices = tensorflow.config.list_physical_devices('GPU')
    print("Num GPUs:", len(physical_devices))
    try:
        spacy.prefer_gpu()
    except:
        print("Not able to activate gpu")
```

```
Num GPUs: 1
```

Hi reader, if you wish to optimize performence more you can read the link below

Feel free to read more about optimizations (https://spacy.io/usage/processing-pipelines)

The Data

We will begin with explaining the datasets that we are working with. These are built quite differently.

Data_a

The dataset contains twitter comments with a class column that that gives 1 if there is offensive language, 0 if there is hate speech and 2 if there is neither.

Data_b

The dataset is from wikipedia texts classifies whether each text is toxic speech or threatining speech or other types, and we were able to take that and say that if any of those classifications exist that we can label it as profane language.

Data_c

We will also use data from sources that were not manually labeled as part of a sponsored project (Keggle/etc..), this type of data is generated from known sources with high rate of success being correct without manual verification. We used scrappy opensource package to crawl "https://www.goodnewsnetwork.org/" (https://www.goodnewsnetwork.org/") and extract the text from the articles that were posted there.

Data_d, e, f, g, h

Random lists of bad words we found online that come in different formats

Data - Source URLS

Labled Datasets

Database 1 - Source (https://github.com/t-davidson/hate-speech-and-offensive-language/tree/master/data)

Database 2 - Source (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/)

Scrapy

The good news network website (https://www.goodnewsnetwork.org/more/about-us/)

Bad words lists

DB4 (https://github.com/web-mech/badwords/blob/master/lib/lang.json) DB5 (http://www.bannedwordlist.com/) DB6 (https://www.freewebheaders.com/bad-words-list-and-page-moderation-words-list-for-facebook/) DB7 (https://www.freewebheaders.com/youtube-blacklist-words-list-youtube-comment-moderation/) DB8 (https://www.freewebheaders.com/full-list-of-bad-words-banned-by-google/)

Handling the data

After we gather the datasets that we want to use we have to clean the text of any superfluous characters that will not help us with detremining the sentiment of the sentence. The template should be some how similiar to "yes \ no" of whether the text (sentences) are offensive or not.

So the goal is that for each row we shall provide a binary indexing of "Offensive" or not. To have a uniform text template we shall use filtering techniques such as splits of the parahraph to sentences, tokenizations, characters removals and more. In addition we shall add another column of the words counts. This can help with determing the "weight" of the word on the sentence

Before handling the data, below you can find the Data_C scrappy code below

Files - Format is ("file_name.py", path)

"items.py" file --> scrapygoodnews\scrapygoodnews\items.py

In [4]:
```
import scrapy

class ScrapygoodnewsItem(scrapy.Item):
    story = scrapy.Field()
    url = scrapy.Field()
```

"goodnews_scrape.py" file --> scrapygoodnews\scrapygoodnews\spiders\goodnews_scrape.py

In [5]:
```
import scrapy
from scrapygoodnews.scrapygoodnews.items import ScrapygoodnewsItem

class Goodnews(scrapy.Spider):
    name = "my_scraper"
    custom_settings = {
        'FEEDS': {
            'scrapygoodnews\scrapygoodnews\output\stories.csv': {
                'format': 'csv',
                'overwrite': True
            }
        }}

    allowed_domains = ['www.goodnewsnetwork.org']
    # First Start Url
    start_urls = ["https://www.goodnewsnetwork.org/category/news/page/1/"]
    n_pages = 10**7

    for i in range(2, n_pages):
        start_urls.append("https://www.goodnewsnetwork.org/category/news/page/" + str(i))

    def parse(self, response):
        for href in response.xpath(
                '//h3[@class="entry-title td-module-title"]//@href').extract():
            yield scrapy.Request(href, callback=self.parse_dir_contents)

    def parse_dir_contents(self, response):
        item = ScrapygoodnewsItem()

        # Getting Story
        story_list = response.xpath('//div[@class="td-post-content"]//p/text()').extract()
        story_list = [x.strip() for x in story_list if len(x.strip()) > 0]

        if len(story_list) > 0:
            item['story'] = " ".join(story_list)# Url (The link to the page)
            item['url'] = response.xpath("//meta[@property='og:url']/@content").extract()
            yield item
        else:
            pass
```

settings.py --> \scrapygoodnews\scrapygoodnews\settings.py

In [6]:
```
# Partial snippet
"""
USER_AGENT = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari
LOG_LEVEL = logging.WARNING
COOKIES_ENABLED = False
TELNETCONSOLE_ENABLED = False
"""
print()
```

run_spider_file.py --> \scrapygoodnews\scrapygoodnews\run_spider_file.py

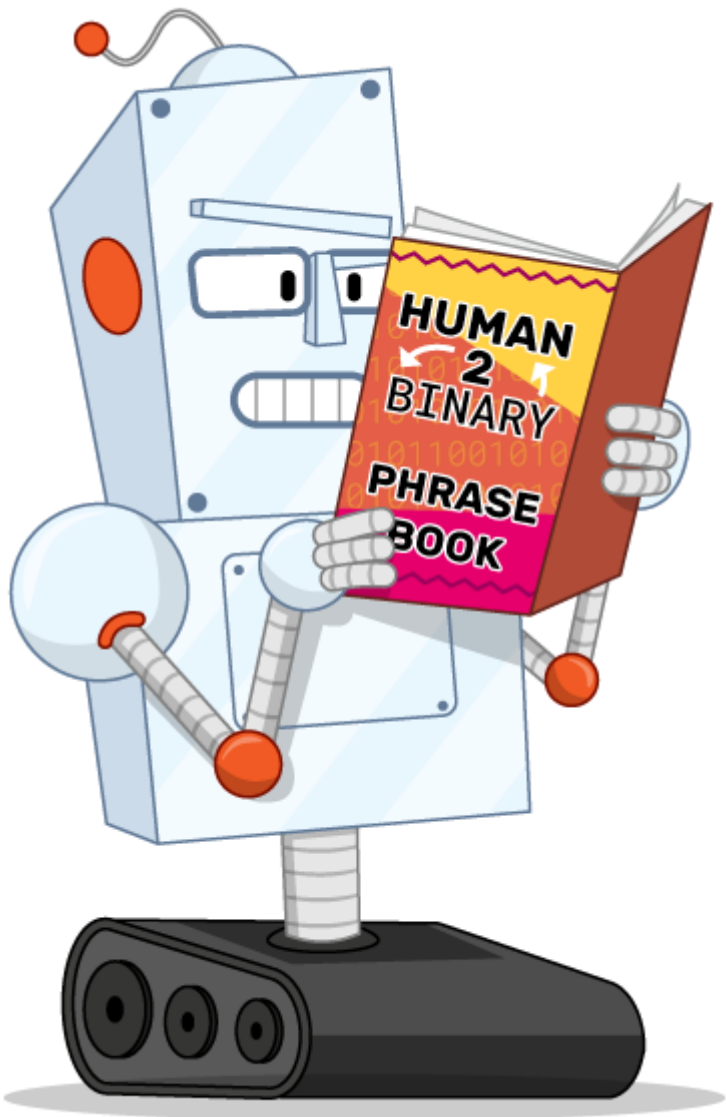
In [7]:
```
from scrapygoodnews.scrapygoodnews.spiders.goodnews_scrape import Goodnews
import scrapygoodnews.scrapygoodnews.settings as my_settings
from scrapy.settings import Settings
from scrapy.crawler import CrawlerProcess

stop_after_crawl = True

def run_spider():
    """run spider with Goodnews"""
    # Import settings from project and not terminal default path
    crawler_settings = Settings()
    crawler_settings.setmodule(my_settings)

    crawler = CrawlerProcess(crawler_settings)
    # Avoid Twisted reactor issue - For running the same notebook
    print("Spider start running\n /\/\\(ر •`•ر)/\\/\\ \t /\/\\(ر •`•ر)/\/\\ \t /\/\\(ر •`•ر)/\\/\\")
    crawler.crawl(Goodnews)
    crawler.start(stop_after_crawl=stop_after_crawl)
    print("Spider end")
```

Crawl - Set this parameter to "True" or "False" if you wish to activate it from your Jupyter notebook

In [8]:
```
activate_crawl = True
```

In [9]:
```
from scrapygoodnews.scrapygoodnews.run_spider_file import run_spider

if activate_crawl:
    run_spider()
```

```
Spider start running
 /\/\(ړ •`•ʖ)/\/\        /\/\(ړ •`•ʖ)/\/\        /\/\(ړ •`•ʖ)/\/\
```



Handling the data - Continue

First we will impot the required packages and load all the data into Pandas framework

In [10]:
```
import pandas as pd
import re
import swifter
from spacy import load
from spacy.tokenizer import Tokenizer
from nltk.tokenize import sent_tokenize, word_tokenize, TweetTokenizer, WhitespaceTokenizer
from nltk.stem import PorterStemmer
import time
```

Database acivation - Set this parameter to "True" or "False" if you wish to activate the filtering

In [11]:
```
activate_db_filtering = True
```

In [12]:
```
if activate_db_filtering:
    data_a = pd.read_csv(r'materials\Cornnel\data\data\labeled_data.csv')
    data_b = pd.read_csv(r'materials\Kaggle\train\train.csv')
    data_c = pd.read_csv(r'scrapygoodnews\scrapygoodnews\output\stories.csv')
    data_d = pd.read_json(r'materials\badwords\word_list_a.json')
    data_e = pd.read_csv(r'materials\badwords\swearWords.csv')
    data_f = pd.read_csv(r'materials\badwords\format_b\facebook-bad-words-list_comma-separated-text-file_2021_01_18.txt'
    data_g = pd.read_csv(r'materials\badwords\format_b\youtube-blacklist-words-list_comma-separated-text-file_2021-01-19
    data_h = pd.read_csv(r'materials\badwords\full-list-of-bad-words_csv-file_2021_01_18\full-list-of-bad-words_csv-file_
else:
    balanced = pd.read_csv(r'output\database\balanced.csv', )
```

The format of the text below is:

1. Have a peak on the data
2. Fine tune it
3. Have another peak on it

Data - A

In [13]:
```
if activate_db_filtering:
    display(data_a.head())
```

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

In [14]:
```
if activate_db_filtering:
    data_a_positive = data_a[data_a["class"]==2]
    data_a_positive = pd.DataFrame(data_a_positive["tweet"])
    data_a_positive = data_a_positive.rename(columns={"tweet": "Text"})
    data_a_positive = data_a_positive.assign(Negative=[0 for i in range(len(data_a_positive))])

    data_a_negative = data_a[data_a["class"]!=2]
    data_a_negative = pd.DataFrame(data_a_negative["tweet"])
    data_a_negative = data_a_negative.rename(columns={"tweet": "Text"})
    data_a_negative = data_a_negative.assign(Negative=[1 for i in range(len(data_a_negative))])


    data_a_labeled = pd.concat([data_a_positive, data_a_negative], axis=0)

    tknzr = TweetTokenizer(strip_handles=True, reduce_len=True)

    data_a_labeled['Text'] = data_a_labeled['Text'].swifter.apply(tknzr.tokenize)
    data_a_labeled['Text'] = data_a_labeled['Text'].swifter.apply(" ".join)

    data_a_labeled["Text"] = data_a_labeled["Text"].str.replace('\n', ' ', regex=False).str.lower()
    data_a_labeled["Text"] = data_a_labeled["Text"].str.replace(r'("|! ! ! rt :)' , '', regex=True)
    data_a_labeled["Text"] = data_a_labeled["Text"].str.replace(r'(rt : )' , '', regex=True)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [15]:
```
if activate_db_filtering:
    display(data_a_labeled)
```

	Text	Negative
0	as a woman you shouldn't complain about clean...	0
40	momma said no pussy cats inside my doghouse	0
63	: - simplyaddictedtoguys http://t.co/1jl4hi8z...	0
66	: http://t.co/3gzupfumev woof woof and hot s...	0
67	: lemmie eat a oreo & do these dishes . one ...	0
...	...	...
24776	you're all niggers	1
24777	you're such a retard i hope you get type 2 dia...	1
24778	you's a muthaf * * * in lie " : right ! his tl...	1
24780	young buck wanna eat ! ! .. dat nigguh like i ...	1
24781	youu got wild bitches tellin you lies	1

24783 rows × 2 columns

Data - B

In [16]:
```
if activate_db_filtering:
    display(data_b.head())
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

In [17]:
```
if activate_db_filtering:
    data_b_positive = data_b.loc[(data_b['toxic']==0) & (data_b['severe_toxic']==0) & (data_b['obscene']==0) & (data_b['
                                 & (data_b['insult']==0) & (data_b['identity_hate']==0)]

    data_b_positive = pd.DataFrame(data_b_positive["comment_text"])
    data_b_positive = data_b_positive.rename(columns={"comment_text": "Text"})
    data_b_positive = data_b_positive.assign(Negative=[0 for i in range(len(data_b_positive))])

    data_b_negative = data_b.loc[(data_b['toxic']==1) | (data_b['severe_toxic']==1) | (data_b['obscene']==1) | (data_b['
                                 | (data_b['insult']==1) | (data_b['identity_hate']==1)]
    data_b_negative = pd.DataFrame(data_b_negative["comment_text"])
    data_b_negative = data_b_negative.rename(columns={"comment_text": "Text"})
    data_b_negative = data_b_negative.assign(Negative=[1 for i in range(len(data_b_negative))])


    data_b_labeled = pd.concat([data_b_positive, data_b_negative], axis=0)
    data_b_labeled["Text"] = data_b_labeled["Text"].str.replace('\n', ' ', regex=False).str.lower()
    data_b_labeled["Text"] = data_b_labeled["Text"].str.replace('"', '', regex=False)
```

In [18]:
```
if activate_db_filtering:
    display(data_b_labeled)
```

	Text	Negative
0	explanation why the edits made under my userna...	0
1	d'aww! he matches this background colour i'm s...	0
2	hey man, i'm really not trying to edit war. it...	0
3	more i can't make any real suggestions on imp...	0
4	you, sir, are my hero. any chance you remember...	0
...	...	...
159494	our previous conversation you fucking shi...	1
159514	you are a mischievious pubic hair	1
159541	your absurd edits your absurd edits on great...	1
159546	hey listen don't you ever!!!! delete my edit...	1
159554	and i'm going to keep posting the stuff u dele...	1

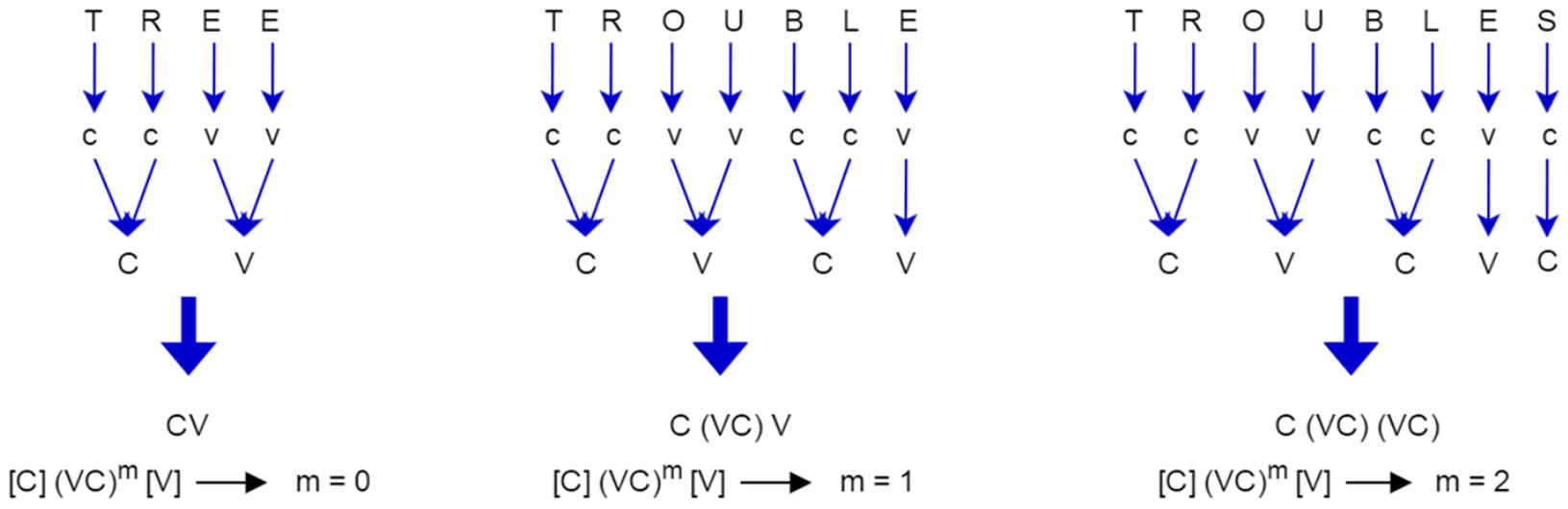
159571 rows × 2 columns

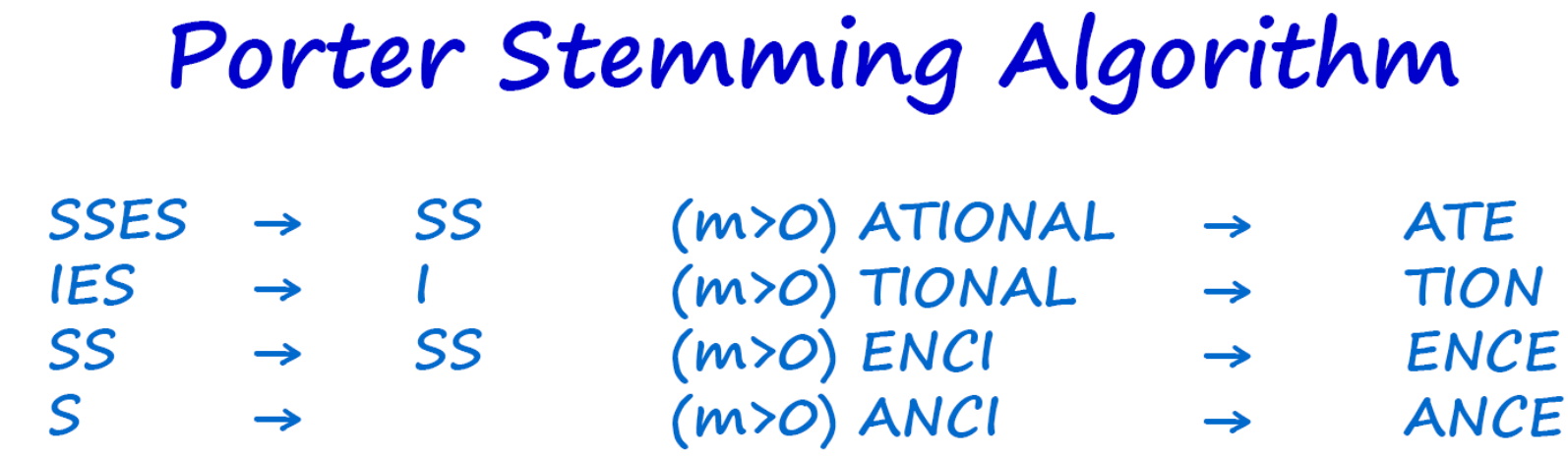
Data - C

In [19]:
```
if activate_db_filtering:
    display(data_c.head())
```

	story	url
0	A landfill of 42 million tires in the sands of...	https://www.goodnewsnetwork.org/kuwait-tire-mo...
1	If you feel like you’re hoarding plastic conta...	https://www.goodnewsnetwork.org/americans-hoar...
2	A design for plastic bottles that can be linke...	https://www.goodnewsnetwork.org/friendship-bot...
3	Manchester City are the reigning champions of ...	https://www.goodnewsnetwork.org/manchester-cit...
4	There may be a thousand ways to say, ‘Happy Bi...	https://www.goodnewsnetwork.org/restaurant-bra...

We can use porter stemming to reduce the complexity, I eventually chose not to use it





porter stemming acivation - Set this parameter to "True" or "False" if you wish to activate the filtering

In [20]:
```
porter_filtering = False
```

In [21]:
```
if activate_db_filtering:
    data_c_story = pd.DataFrame(data_c["story"])
    splitted_data = []

    if porter_filtering:
        porter = PorterStemmer()
        for text in data_c_story["story"]:
            splitted_sent = sent_tokenize(text)
            for sent in splitted_sent:
                token_words = word_tokenize(sent)
                portered = [porter.stem(word) for word in token_words]
                splitted_data.append(" ".join(portered))
    else:
        for text in data_c_story["story"]:
            splitted = sent_tokenize(text)
            for i in splitted:
                splitted_data.append(i)

    splitted_data = pd.DataFrame(splitted_data, columns=["Text"])
    data_c_labeled = splitted_data.assign(Negative=[0 for i in range(len(splitted_data))])
```

In [22]:
```
if activate_db_filtering:
    display(data_c_labeled)
```

	Text	Negative
0	A landfill of 42 million tires in the sands of...	0
1	This news in itself would be a major relief to...	0
2	But the government isn’t stopping there.	0
3	They are aiming to create a green city of 25,0...	0
4	The first step is to clear the ground.	0
...	...	...
9586	A generous couple has been secretly stuffing m...	0
9587	Krystal Duhaney is a registered nurse and the ...	0
9588	A soon-to-be mother of three, when she and her...	0
9589	Now they’re in a better place financially, the...	0
9590	“We recalled how hard it was for us as new par...	0

9591 rows × 2 columns

More negative word list - Data D to Data H

In [23]:
```
if activate_db_filtering:
    display(data_d.head())
```

	words
0	ahole
1	anus
2	ash0le
3	ash0les
4	asholes

In [24]:
```
if activate_db_filtering:
    data_d_labeled = data_d.rename(columns={"words": "Text"})
    data_d_labeled = data_d_labeled.assign(Negative=[1 for i in range(len(data_d_labeled))])
    data_d_labeled
```

In [25]:
```
if activate_db_filtering:
    display(data_e.head())
```

anal	anus	arse	ass	ballsack	balls	bastard	bitch	biatch	bloody	...	smegma	spunk	tit	tosser	turd	twat	vagina	wank	whore	wtf

0 rows × 77 columns

In [26]:
```
if activate_db_filtering:
    data_e_labeled = [[i, 1] for i in data_e]
    data_e_labeled = pd.DataFrame(data_e_labeled, columns = ["Text", "Negative"])
```

In [27]:
```
if activate_db_filtering:
    display(data_e_labeled)
```

	Text	Negative
0	anal	1
1	anus	1
2	arse	1
3	ass	1
4	ballsack	1
...	...	...
72	twat	1
73	vagina	1
74	wank	1
75	whore	1
76	wtf	1

77 rows × 2 columns

In [28]:
```
if activate_db_filtering:
    display(data_f.tail())
```

	## Facebook Page Moderation Words List (Comma Separated Text File)
4	## URL: https://www.freewebheaders.com/bad-wor...
5	## Copy all the words below:
6	----------
7	4r5e, 5h1t, 5hit, a55, anal, anus, ar5e, arrse...
8	----------...

In [29]:
```
if activate_db_filtering:
    data_f_labeled = data_f.iloc[7]
    data_f_labeled = [[i,1] for i in data_f_labeled[0].split(",")]
    data_f_labeled = pd.DataFrame(data_f_labeled, columns = ["Text", "Negative"])
```

In [30]:
```
if activate_db_filtering:
    display(data_f_labeled)
```

	Text	Negative
0	4r5e	1
1	5h1t	1
2	5hit	1
3	a55	1
4	anal	1
...	...	...
1008	xx	1
1009	yaoi	1
1010	yellow showers	1
1011	yiffy	1
1012	zoophilia	1

1013 rows × 2 columns

In [31]:
```
if activate_db_filtering:
    display(data_g.tail())
```

	## Youtube Blacklist Words List (Comma-separated-Text-File)
3	## URL: https://www.freewebheaders.com/youtube...
4	## Copy all the words below:
5	## ----------
6	2 girls 1 cup, 2g1c, 4r5e, 5h1t, 5hit, a$$, a$...
7	----------...

In [32]:
```
if activate_db_filtering:
    data_g_labeled = data_g.iloc[6]
    data_g_labeled = [[i,1] for i in data_g_labeled[0].split(",")]
    data_g_labeled = pd.DataFrame(data_g_labeled, columns = ["Text", "Negative"])
```

In [33]:
```
if activate_db_filtering:
    display(data_g_labeled)
```

	Text	Negative
0	2 girls 1 cup	1
1	2g1c	1
2	4r5e	1
3	5h1t	1
4	5hit	1
...	...	...
3461	yiffy	1
3462	yobbo	1
3463	zoophile	1
3464	zoophilia	1
3465	zubb	1

3466 rows × 2 columns

In [34]:
```
if activate_db_filtering:
    display(data_h.head())
```

	2 girls 1 cup	Unnamed: 1	Unnamed: 2	## Full List of Bad Words (CSV File)
0	2g1c	NaN	NaN	==============================
1	4r5e	NaN	NaN	NaN
2	5h1t	NaN	NaN	## This Full List of Words is provided free b...
3	5hit	NaN	NaN	NaN
4	a$$	NaN	NaN	## Last Update: Jan 18, 2021

In [35]:
```
if activate_db_filtering:
    data_h_labeled = data_h.iloc[:, 0]
    data_h_labeled = [[i, 1] for i in data_h_labeled]
    data_h_labeled = pd.DataFrame(data_h_labeled, columns = ["Text", "Negative"])
```

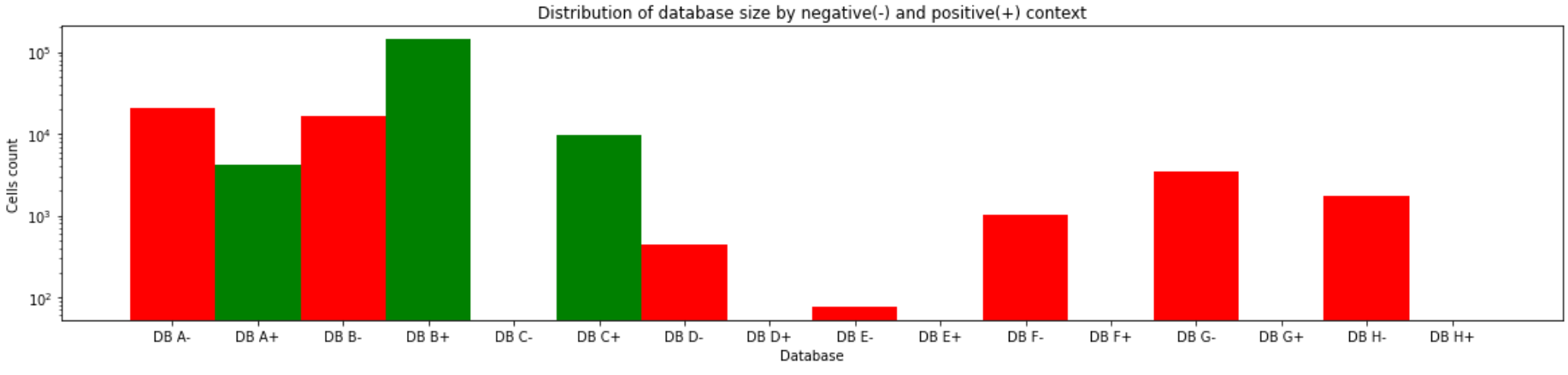
In [36]:
```
if activate_db_filtering:
    display(data_h_labeled)
```

	Text	Negative
0	2g1c	1
1	4r5e	1
2	5h1t	1
3	5hit	1
4	a$$	1
...	...	...
1728	yiffy	1
1729	yobbo	1
1730	zoophile	1
1731	zoophilia	1
1732	zubb	1

1733 rows × 2 columns

Concating all the data from all different sources into 1 uniformed data frame

In [37]:
```
if activate_db_filtering:
    data_dis = pd.DataFrame([len(data_a_labeled[data_a_labeled["Negative"]==1]), len(data_a_labeled[data_a_labeled["Negat
    plt_x_axis = []
    color_axis = []
    [[color_axis.append("Red"), color_axis.append("Green"), plt_x_axis.append("DB "+chr(65+i)+"-"),plt_x_axis.append("DB
    plt.figure(figsize=(20, 4))
    plt.bar(plt_x_axis, data_dis[0], width=1, log=True, color=color_axis)
    plt.title('Distribution of database size by negative(-) and positive(+) context')
    plt.ylabel("Cells count")
    plt.xlabel("Database")
    print()
else:
    color_axis = ["Red", "Green"]
```

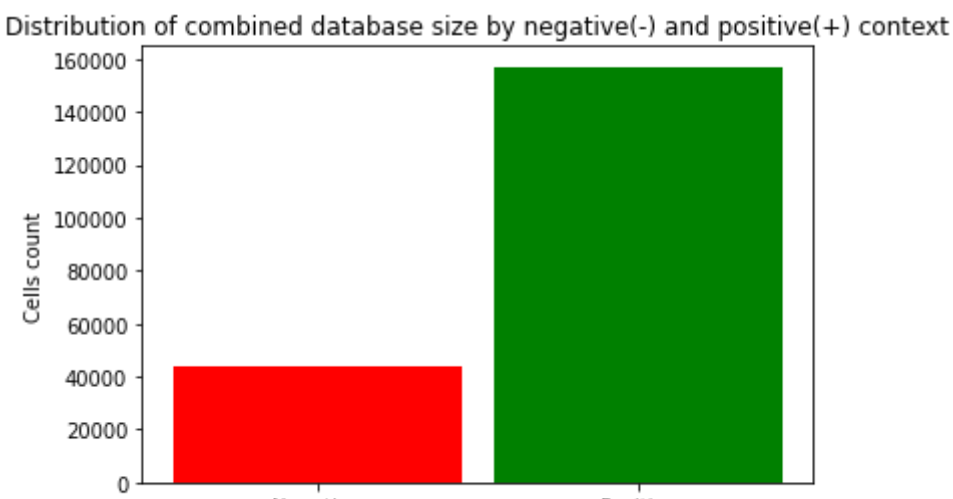


In [38]:
```
if activate_db_filtering:
    data_unfinished = pd.concat([data_a_labeled,data_b_labeled,data_c_labeled, data_d_labeled, data_e_labeled, data_f_lab
```

In [39]:
```
if activate_db_filtering:
    data_unfinished[data_unfinished["Negative"]==1]
```

In [40]:
```
if activate_db_filtering:
    data_unfinished[data_unfinished["Negative"]==0]
```

In [41]:
```
if activate_db_filtering:
    neg = len(data_unfinished[data_unfinished["Negative"]==1])
    pos = len(data_unfinished[data_unfinished["Negative"]==0])
else:
    neg = len(balanced[balanced["Negative"]==1])
    pos = len(balanced[balanced["Negative"]==0])

ratio = min(neg, pos)/max(neg, pos)

plt.bar(["Negative","Positive"], [neg, pos], width=0.9, color=color_axis)
plt.title('Distribution of combined database size by negative(-) and positive(+) context')
plt.ylabel("Cells count")
plt.xlabel("Ratio:  1 to {}".format(round(ratio,3)))
print()
```



There are, unsuprisingly due to our data sources, more positive sentences than offensive.

We can, by design, split the data to have the same number of positive and negative rows. We will also shuffel the rows to have a uniformed data spread along the sheet.

Split - set this parameter to "True" or "False" if you wish to have balanced negative and positive sentences

In [42]:
```
# Enable this to have 50%/50% of negative and positive
split_data = True
```

In [43]:
```
if activate_db_filtering:
    if split_data:
        if neg>pos:
            negative_finished = data_unfinished[data_unfinished["Negative"]==1].sample(frac = ratio)
            positive_finished = data_unfinished[data_unfinished["Negative"]==0]
        else:
            positive_finished = data_unfinished[data_unfinished["Negative"]==0].sample(frac = ratio)
            negative_finished = data_unfinished[data_unfinished["Negative"]==1]

        # Concat 50/50 datasets
        balanced = pd.concat([positive_finished,negative_finished], axis=0)
        # Shuffle the rows
        balanced = balanced.sample(frac = 1).reset_index(drop=True)
        data_unbalanced = data_unfinished.sample(frac = 1).reset_index(drop=True)
    else:
        data_unbalanced = data_unfinished.sample(frac = 1).reset_index(drop=True)
        balanced = data_unbalanced
```

In [44]:
```
display(balanced[balanced["Negative"]==0])
```

	Text	Negative
0	hadith i think there should be hadith fro...	0
1	garbage, selective pieces of news, not the who...	0
2	more than anything else, common sense is req...	0
5	interlinear interpolations such as this: usua...	0
8	hello, muthurajalm, and welcome to wikipedia! ...	0
...	...	...
87151	i did find an image on line of a monumental s...	0
87152	you hey bro why do you love grace saunder so m...	0
87157	falcom and magluta hi there.. two things i nee...	0
87160	{unblock|i apparently have been sharing my ip ...	0
87161	agreed. and this is over 6 years later...tha...	0

43581 rows × 2 columns

In [45]:
```
display(balanced[balanced["Negative"]==1])
```

	Text	Negative
3	mcfagget	1
4	you gotta be kidding you arrogant sob!	1
6	thanks nig 😜	1
7	ok now you're talking my language , none of th...	1
12	be a girl with a mind , a bitch with an attitu...	1
...	...	...
87154	element number 35 is bromine and it comes befo...	1
87155	so... are you this arrogant and gutless in re...	1
87156	i don't bend that way queer . thanks for the o...	1
87158	omfg or the other pics of the tranny lmao	1
87159	faggot fuck you bitch you're a faggot	1

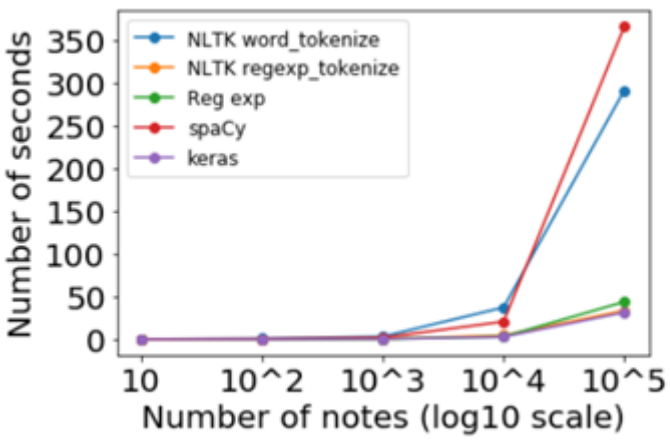
43581 rows × 2 columns

In [46]:
```
display(balanced.tail())
```

	Text	Negative
87157	falcom and magluta hi there.. two things i nee...	0
87158	omfg or the other pics of the tranny lmao	1
87159	faggot fuck you bitch you're a faggot	1
87160	{unblock|i apparently have been sharing my ip ...	0
87161	agreed. and this is over 6 years later...tha...	0

Eventually we will tokenize the data with the same common tokenizer.

To give a sense of generalization to the data and make it even more uniformed we add an optional feature to transform the data once again with tokenizer. We chose to use NLTK regex as a our final tokenizer and not other due to its speed.



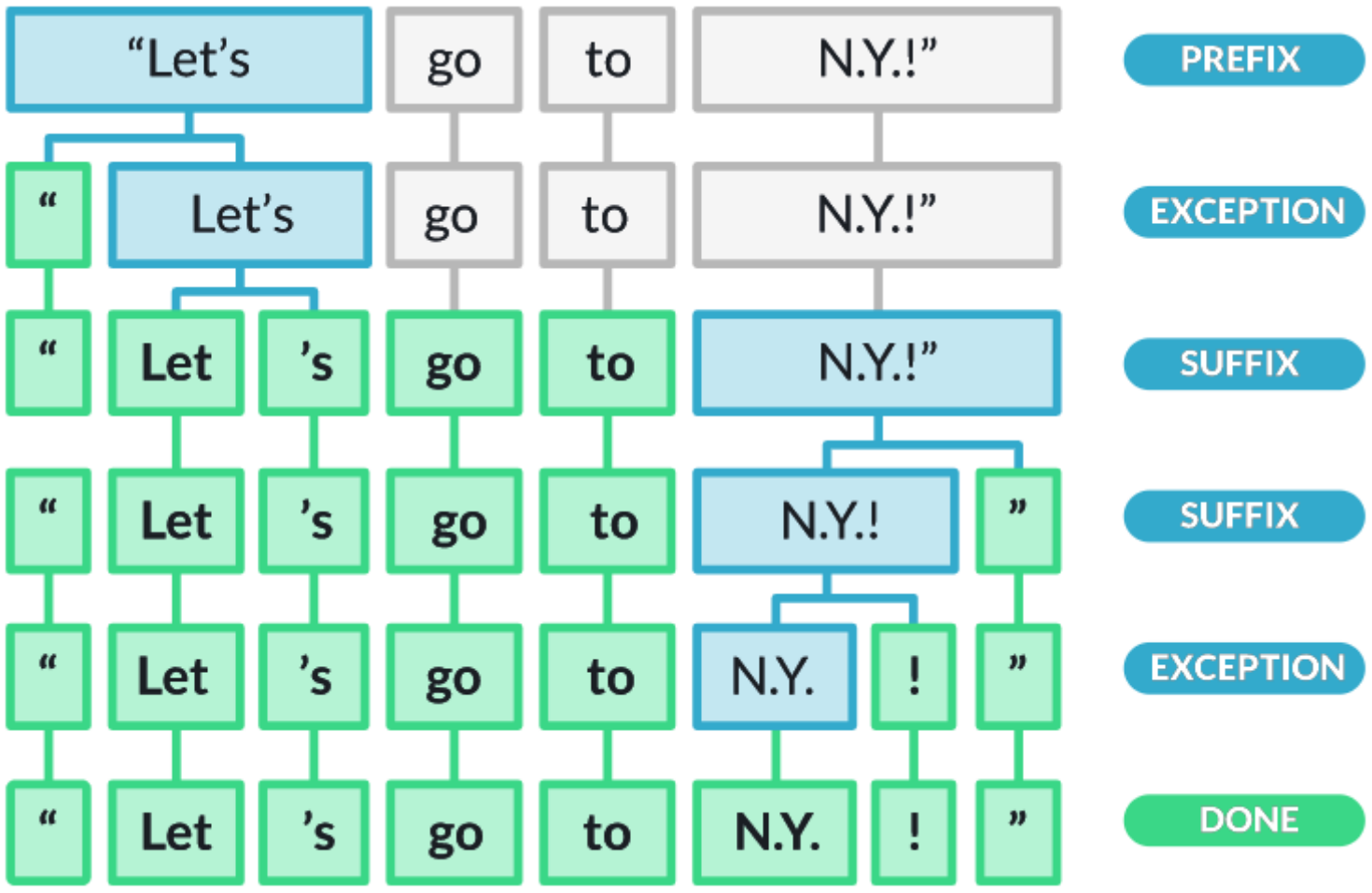
Alterntaive option is to use Spacy as a our final tokenizer and not NLTK. This is because while NLTK tokenizers can have a better taylor made solutions, Spacy tokenizer has a better "Single point solution" that generally suits all text sources.

It is important to note that usually any kind of generalization can reduce the accuracy of our model. In other words, our method will not provide the best reulsts but because the field of tokenization can have a whole notebook of itself we do not want to waste major time on that. Our point here is to enable high flexability for our datasets. Anu user might choose to add or remove some of sets in the future and will not need to customize a lot of the code.

Also, as a side note, for dataset "A" we used NLTK tweeter custom made tokenizer. For dataset "B" we used the sentences NLTK tokenizer that provides faster tokenization.

Because of the Spacy DependencyParser, the operation takes some time. If you run it, let the computer work for a while you drink a cup of coffee a refresh ;)

Read more about Spacy here (https://spacy.io/)



Accuracy Evaluation ¶

TOKEN_ACC	Tokenization	1.00
TAG_ACC	Part-of-speech tags (fine grained tags, Token.tag)	0.97
DEP_UAS	Unlabelled dependencies	0.92
DEP_LAS	Labelled dependencies	0.90
ENTS_P	Named entities (precision)	0.84
ENTS_R	Named entities (recall)	0.83
ENTS_F	Named entities (F-score)	0.84
SENTS_P	Sentence segmentation (precision)	0.91
SENTS_R	Sentence segmentation (recall)	0.88
SENTS_F	Sentence segmentation (F-score)	0.89

spacy_enabled - set True or False the parameter below for activation

In [47]:
```
nltk_regex = True
spacy_enabled = not nltk_regex
```

In [48]:
```
if activate_db_filtering:
    balanced["Text"] = balanced["Text"].astype(str)
    if spacy_enabled:
        balanced["Text"] = balanced["Text"].str.replace('  ', ' ', regex=False)
        try:
            sp = spacy.load('en_core_web_sm')
        except:
            !python -m spacy download en_core_web_sm

        sp = spacy.load('en_core_web_sm')

        sent = list(sp.pipe(balanced["Text"]))
        balanced = balanced.assign(Tokenized=sent)

        sentences = [" ".join(row.sents) for row in sent]
        balanced = balanced.assign(sentences=sentences)

    elif nltk_regex:
        ws_tokenize = WhitespaceTokenizer()
        balanced['sentences'] = balanced['Text'].swifter.apply(ws_tokenize.tokenize)
        balanced['sentences'] = balanced['sentences'].swifter.apply(" ".join)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [49]:
```
display(balanced)
```

	Text	Negative	sentences
0	hadith i think there should be hadith fro...	0	hadith i think there should be hadith from buk...
1	garbage, selective pieces of news, not the who...	0	garbage, selective pieces of news, not the who...
2	more than anything else, common sense is req...	0	more than anything else, common sense is requi...
3	mcfagget	1	mcfagget
4	you gotta be kidding you arrogant sob!	1	you gotta be kidding you arrogant sob!
...	...	...	...
87157	falcom and magluta hi there.. two things i nee...	0	falcom and magluta hi there.. two things i nee...
87158	omfg or the other pics of the tranny lmao	1	omfg or the other pics of the tranny lmao
87159	faggot fuck you bitch you're a faggot	1	faggot fuck you bitch you're a faggot
87160	{unblock|i apparently have been sharing my ip ...	0	{unblock|i apparently have been sharing my ip ...
87161	agreed. and this is over 6 years later...tha...	0	agreed. and this is over 6 years later...that ...

87162 rows × 3 columns

We can also give weight, per word, for the sentence meaning.

We must surely understand by now that "Fuck", a 1 word curse said alone, clearly has a negative meaning. While other sentences, such as: "What the fuck just happend", has slighly less negative meaning.

Lets try to give these sentences weight by the inverse of the number of words

weight_per_word - set True or False the parameter below for activation

In [50]:
```
weight_per_word = True
```

In [51]:
```
if weight_per_word:
    # We will seperate by the word counts
    pattern = re.compile(r'\w+')
    balanced['Number of words'] = balanced['sentences'].swifter.apply(lambda x: max(1, len(pattern.findall(x))))
    balanced['Weight per word'] = balanced['Number of words'].swifter.apply(lambda x: 1/x)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [52]:
```
display(balanced.reset_index(drop=True))
```

	Text	Negative	sentences	Number of words	Weight per word
0	hadith i think there should be hadith fro...	0	hadith i think there should be hadith from buk...	24	0.041667
1	garbage, selective pieces of news, not the who...	0	garbage, selective pieces of news, not the who...	29	0.034483
2	more than anything else, common sense is req...	0	more than anything else, common sense is requi...	194	0.005155
3	mcfagget	1	mcfagget	1	1.000000
4	you gotta be kidding you arrogant sob!	1	you gotta be kidding you arrogant sob!	7	0.142857
...	...	...	...	...	...
87157	falcom and magluta hi there.. two things i nee...	0	falcom and magluta hi there.. two things i nee...	112	0.008929
87158	omfg or the other pics of the tranny lmao	1	omfg or the other pics of the tranny lmao	9	0.111111
87159	faggot fuck you bitch you're a faggot	1	faggot fuck you bitch you're a faggot	8	0.125000
87160	{unblock|i apparently have been sharing my ip ...	0	{unblock|i apparently have been sharing my ip ...	43	0.023256
87161	agreed. and this is over 6 years later...tha...	0	agreed. and this is over 6 years later...that ...	16	0.062500

87162 rows × 5 columns

save_database - set True or False the parameter below for activation

In [53]:
```
save_database = True
```

In [54]:
```
if save_database:
    balanced.to_csv('output\\database\\balanced.csv', index=False)
```

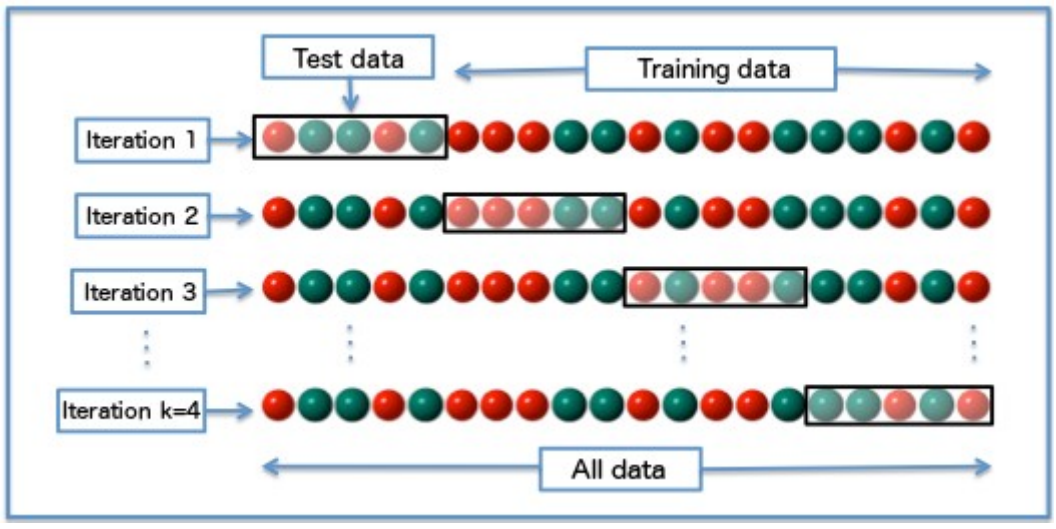
In [55]:
```
balanced.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87162 entries, 0 to 87161
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Text             87162 non-null  object 
 1   Negative         87162 non-null  int64  
 2   sentences        87162 non-null  object 
 3   Number of words  87162 non-null  int64  
 4   Weight per word  87162 non-null  float64
dtypes: float64(1), int64(2), object(2)
memory usage: 3.3+ MB
```

The Model

Our model is based on the idea that some words can have more than 1 meaning. How to decide whether a word has a negative or positive context is not an easy task. While some words are clearly offensive, some may or may not be offensive. Thus the splitting of the data have a critical effect. One way to overcome overfitting and reach the best results it is to use cross validation.

Also, in our case we have a new, untrained model. We will create our CalibratedClassifierCV. With cv in the parameters as the number of folds. We later fit the model. Because our model is untrained, X and y have to be used for both training and calibration. The way to ensure the data is 'disjoint' is our cross validation: for any given fold, CCCV will split X and y into your training and calibration data, so they do not overlap.



EDA

In [57]:
```
import numpy as np

# Save results
from joblib import dump

# Model evaluation and results
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.calibration import CalibratedClassifierCV

# BoW
from sklearn.feature_extraction.text import TfidfVectorizer

# Models
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import BernoulliNB, MultinomialNB, ComplementNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

import scipy.stats as stats



import seaborn as sns


from keras.preprocessing.text import Tokenizer
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout



import transformers

from PIL import Image
import pytesseract
import cv2
import os

result = {}
```

Method 1 - Models with Bag of words

with n_grams and TfidfVectorizerion

The Bag of Words (BoW) model is the simplest form of text representation in numbers. Like the term itself, we can represent a sentence as a bag of words vector (a string of numbers).

TfidfVectorizerion is CountVectorizer (bag of words) with TfidfTransformer. This means we basically fitst count the number of occurrences for all token and later we normalize it according to the frequencies.

Term Frequent (Tf - CountVectorizer) is a measure of how frequently a term, t, appears in our dataset. Inverse Document Frequency (idf) is a measure of how important a term is. We need the IDF value because computing just the Tf alone is not sufficient to understand the importance of words.

Hence, we see that words like “is”, “this”, “and”, etc., are reduced to values closer to 0 and have little importance; while words like “smart”, “amazing”, etc. are words less frequent, thus with more importance.

	the	red	dog	cat	eats	food
1. the red dog →	1	1	1	0	0	0
2. cat eats dog →	0	0	1	1	1	0
3. dog eats food →	0	0	1	0	1	1
4. red cat eats →	0	1	0	1	1	0

In addition, sentences can be splitted into "N grams", which basically means how many word tokens we take together.

Text	N-gram
Data	1-gram
Great information	2-gram
I am fine	3-gram
Nice to meet you	4-gram

Selecting the N grams range to (1,2), will provide us the following output:

('Bi-grams are cool!') == (['bi', 'grams', 'are', 'cool', 'bi grams', 'grams are', 'are cool'])

Parameters

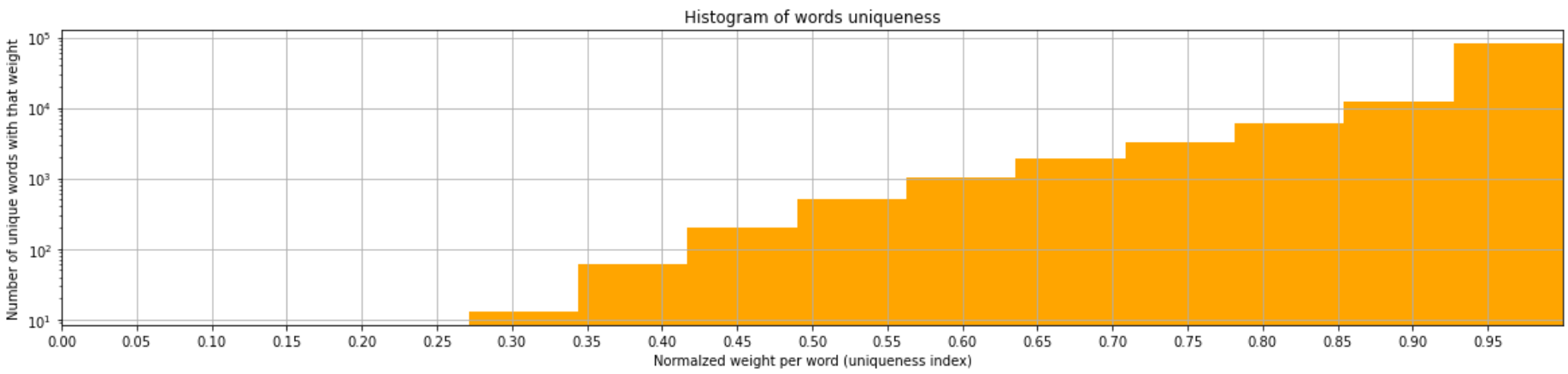
In [58]:
```
# Enable

bow_run = True # True/False to activate/deactivate
save = True
svm = True
bernoulli_bayes = True
multinomial_bayes = True
complement_bayes = True
logistic = True
random_forest = True
neigbors = True
tree = True

neural = True

# Major effect on runtime
ngram_range = (1,1)

max_iter_runtime = 10**5  # For LinearSVC
tolerance = 10**(-2)      # For LinearSVC

neighbors = 10            # K-Neighbors

min_samples_split = 2    # Decision Tree
min_samples_leaf = 1     # Decision Tree

k_fold = 5               # K fold cross validation (CV)

nb_epoch=3
batch_size=32
```

BoW words vector

In [59]:
```
if bow_run:

    X = balanced["sentences"]
    y = balanced['Negative']


    train, test = np.split(balanced, [int(.8*len(balanced))])


    bag_of_words = TfidfVectorizer(stop_words="english", use_idf=True)
    X = bag_of_words.fit_transform(X)

    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,shuffle=False)

    if save:
        dump(bag_of_words, "saved_runs/bag_of_words.joblib")
        dump(X_train, "saved_runs/X.joblib")

    ## Accuracy, Precision, Recall

    classes = ["Positive", "Negative"]
    y_test_array = pd.get_dummies(y_test, drop_first=False).values
```

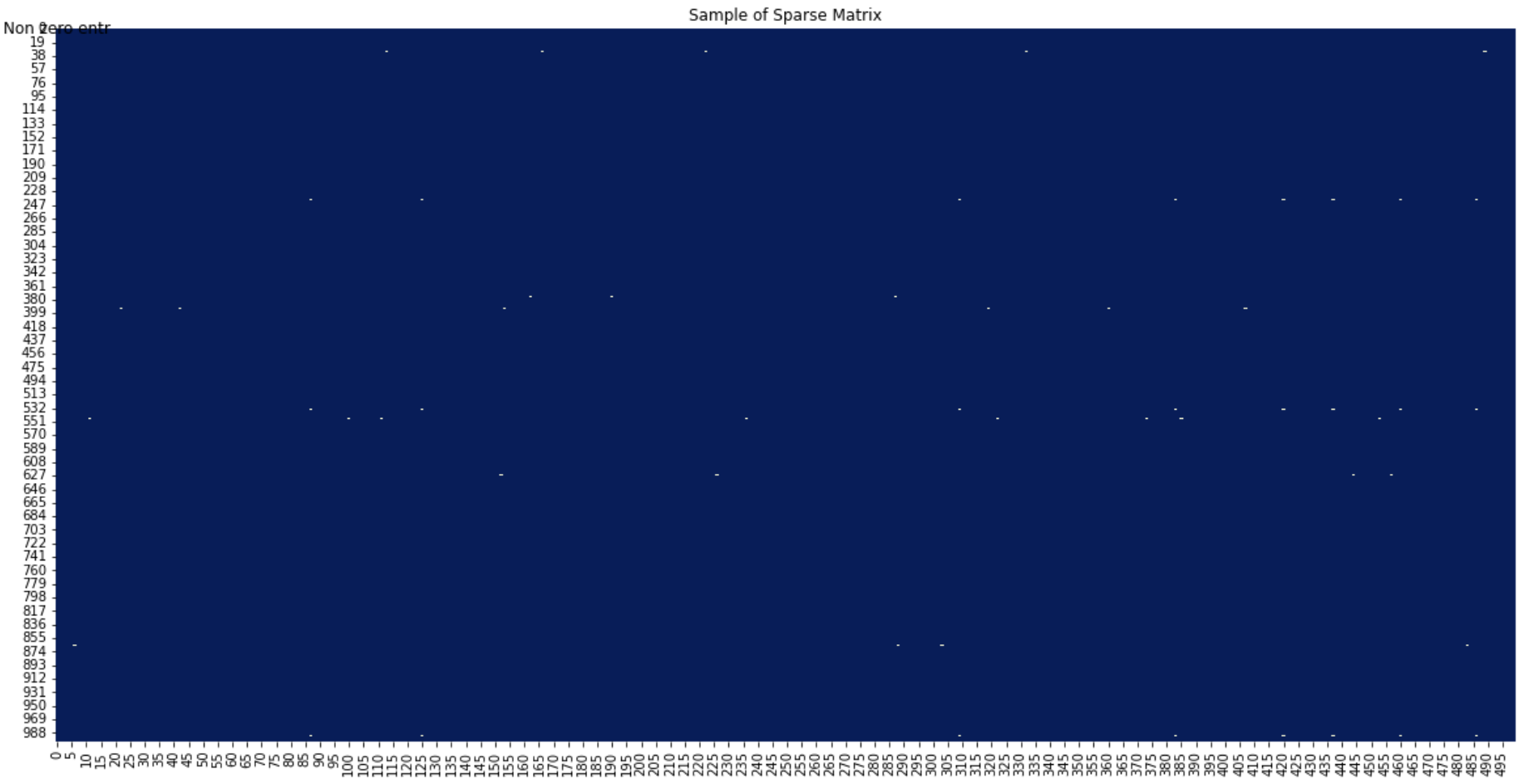
Plotting the words

In [60]:
```
idf_vector = bag_of_words.idf_
idf_vector = idf_vector / np.max(idf_vector)
counts, bins = np.histogram(idf_vector)



figure = plt.figure(figsize=(20, 4))
plt.hist(bins[:-1], bins=bins, weights=counts, color="Orange", log=True)
plt.title('Histogram of words uniqueness')
plt.xlabel('Normalzed weight per word (uniqueness index)')
plt.ylabel('Number of unique words with that weight')
plt.grid(True)
plt.xticks(np.arange(0, 1, step=0.05))
plt.xlim(0, 1)

print()
```



In [68]:
```
figure = plt.figure(figsize=(20, 10))
sns.heatmap(X_train[0:1000].todense()[:,np.random.randint(0,100,500)]==0, vmin=0, vmax=1, cmap="YlGnBu",cbar=False).set_
plt.text(0.5, 0.5, 'Non zero entr', horizontalalignment='center', verticalalignment='center', size="large")
print("White dots are non zero values")
```

```
White dots are non zero values
```



Sparse matrices are those matrices that have the majority of their elements equal to zero. In other words, the sparse matrix can be defined as the matrix that has a greater number of zero elements than the non-zero elements. Storing only the non-zero values and their positions is a common technique in storing sparse data sets and thus avoiding handling a sparse matrix as a dense one which makes excessive use of memory.

Naive Bayes - Bernoulli, Multinomial, Complement . In our case Multinomial.

$$P(y|X) = \frac{P(X|y).\,P(y)}{P(X)}$$

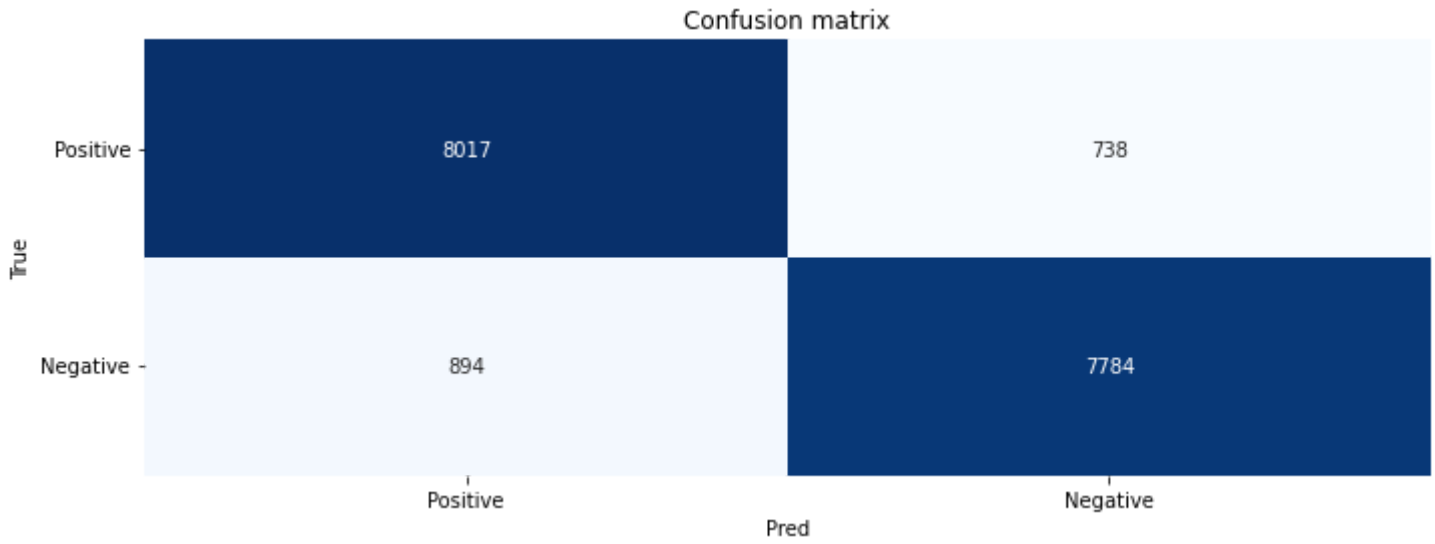
In plain English, this equation is used to answer the following question. “What is the probability of y (my output variable) given X?

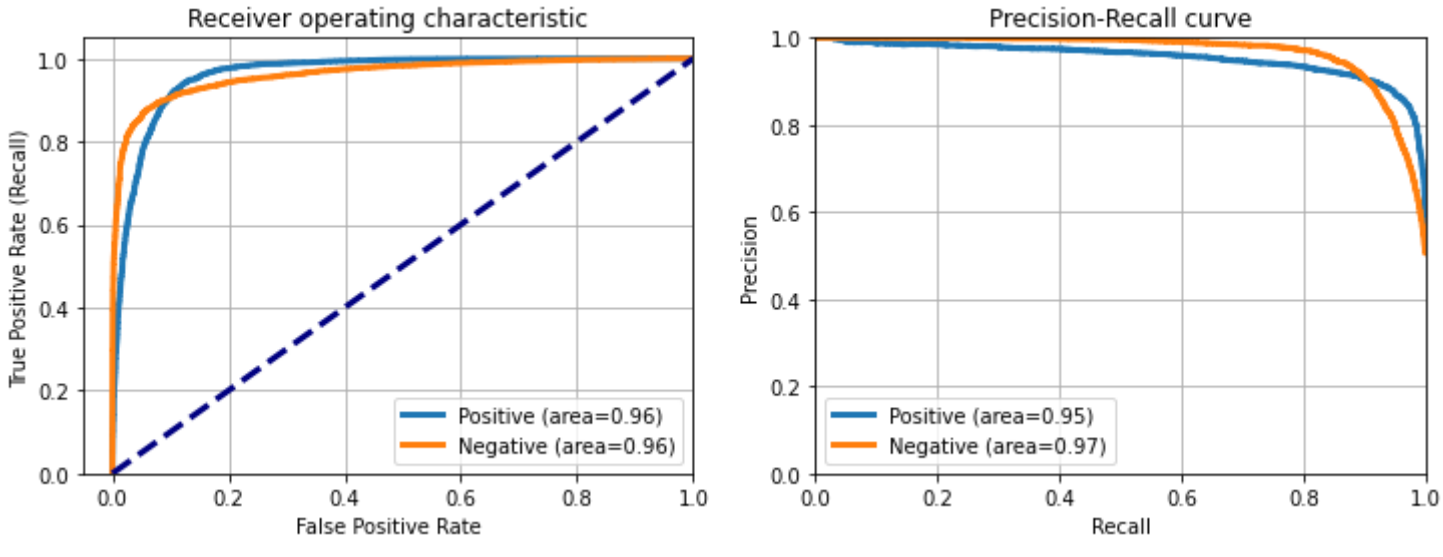
In [69]:
```
if bow_run:
# Naive Bayes
    if bernoulli_bayes:
        model = BernoulliNB()
        calibrated = CalibratedClassifierCV(base_estimator=model,cv=k_fold)
        calibrated.fit(X_train, y_train)
        score = calibrated.score(X_test, y_test)
        result["BernoulliNB"] = score
        print("Score for bernoulli_bayes classifier with fold={} = {} %".format(k_fold, score))
        if save:
            dump(calibrated, "saved_runs/bernoulli_bayes_calibrated.joblib")

    if complement_bayes:
        model = ComplementNB()
        calibrated = CalibratedClassifierCV(base_estimator=model,cv=k_fold)
        calibrated.fit(X_train, y_train)
        score = calibrated.score(X_test, y_test)
        result["ComplementNB"] = score
        print("Score for complement_bayes classifier with fold={} = {} %".format(k_fold, score))
        if save:
            dump(calibrated, "saved_runs/complement_bayes_calibrated.joblib")

    if multinomial_bayes:
        model = MultinomialNB()
        calibrated = CalibratedClassifierCV(base_estimator=model,cv=k_fold)
        calibrated.fit(X_train, y_train)
        score = calibrated.score(X_test, y_test)
        result["MultinomialNB"] = score
        print("Score for multinomial_bayes classifier with fold={} = {} %".format(k_fold, score))
        if save:
            dump(calibrated, "saved_runs/multinomial_bayes_calibrated.joblib")
```

```
Score for bernoulli_bayes classifier with fold=5 = 0.7640107841450123 %
Score for complement_bayes classifier with fold=5 = 0.906384443297195 %
Score for multinomial_bayes classifier with fold=5 = 0.906384443297195 %
```

In [70]:
```
predicted = calibrated.predict(X_test)
predicted_prob = calibrated.predict_proba(X_test)

res1, res2 = map(list, zip(*predicted_prob))

accuracy = metrics.accuracy_score(y_test, predicted)
auc = metrics.roc_auc_score(y_test, res1, multi_class="ovr")
print("Accuracy:{:^20}".format(round(accuracy,2)))
print("Auc:{:^30}".format(round(auc,2)))
print("Detail:")
print(metrics.classification_report(y_test, predicted))

## Plot confusion matrix
cm = metrics.confusion_matrix(y_test, predicted)
fig, ax = plt.subplots()
sns.heatmap(cm, annot=True, fmt='d', ax=ax, cmap=plt.cm.Blues,
            cbar=False)
ax.set(xlabel="Pred", ylabel="True", xticklabels=classes,
       yticklabels=classes, title="Confusion matrix")
plt.yticks(rotation=0)
plt.gcf().set_size_inches(11.5, 4)

fig, ax = plt.subplots(nrows=1, ncols=2)
## Plot roc
for i in range(len(classes)):
    fpr, tpr, thresholds = metrics.roc_curve(y_test_array[:,i],  predicted_prob[:,i])
    ax[0].plot(fpr, tpr, lw=3,
label='{0} (area={1:0.2f})'.format(classes[i],
                                   metrics.auc(fpr, tpr))
                )
ax[0].plot([0,1], [0,1], color='navy', lw=3, linestyle='--')
ax[0].set(xlim=[-0.05,1.0], ylim=[0.0,1.05],
          xlabel='False Positive Rate',
          ylabel="True Positive Rate (Recall)",
          title="Receiver operating characteristic")
ax[0].legend(loc="lower right")
ax[0].grid(True)

## Plot precision-recall curve
for i in range(len(classes)):
    precision, recall, thresholds = metrics.precision_recall_curve(
                 y_test_array[:,i], predicted_prob[:,i])
    ax[1].plot(recall, precision, lw=3,
               label='{0} (area={1:0.2f})'.format(classes[i],
                                  metrics.auc(recall, precision))
              )
ax[1].set(xlim=[0.0,1.05], ylim=[0.0,1.05], xlabel='Recall',
          ylabel="Precision", title="Precision-Recall curve")
ax[1].legend(loc="best")
ax[1].grid(True)
plt.gcf().set_size_inches(12, 4)
plt.xlim(0, 1)
plt.ylim(0, 1)
print(end="")
```

```
Accuracy:        0.91
Auc:             0.04
Detail:
              precision    recall  f1-score   support

           0       0.90      0.92      0.91      8755
           1       0.91      0.90      0.91      8678

    accuracy                           0.91     17433
   macro avg       0.91      0.91      0.91     17433
weighted avg       0.91      0.91      0.91     17433
```

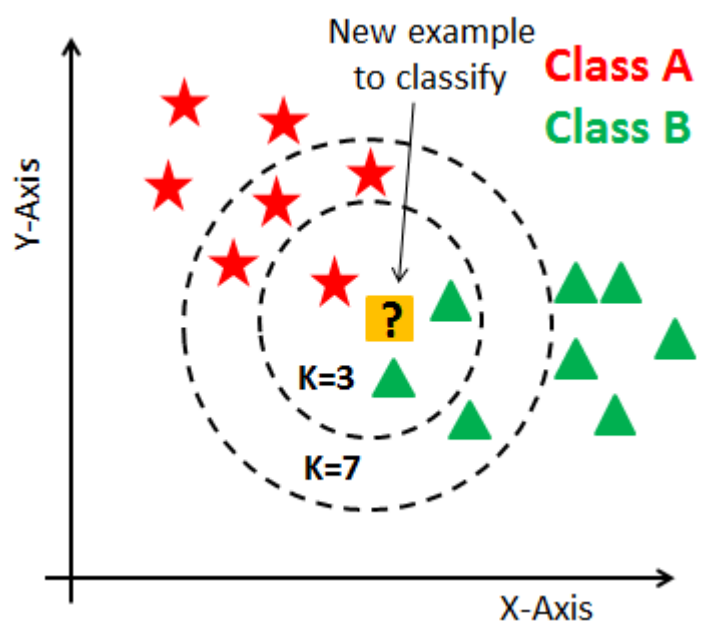




K-Neighbors classifier

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other

As we increase the value of K, that is the number of neihbors, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far

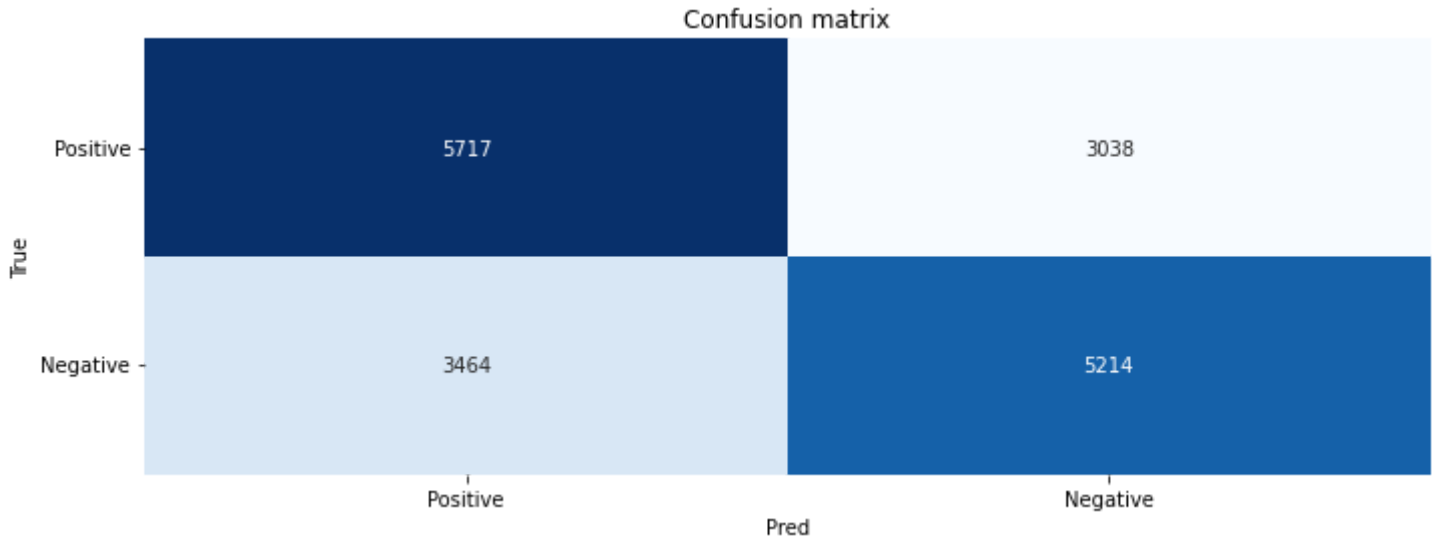


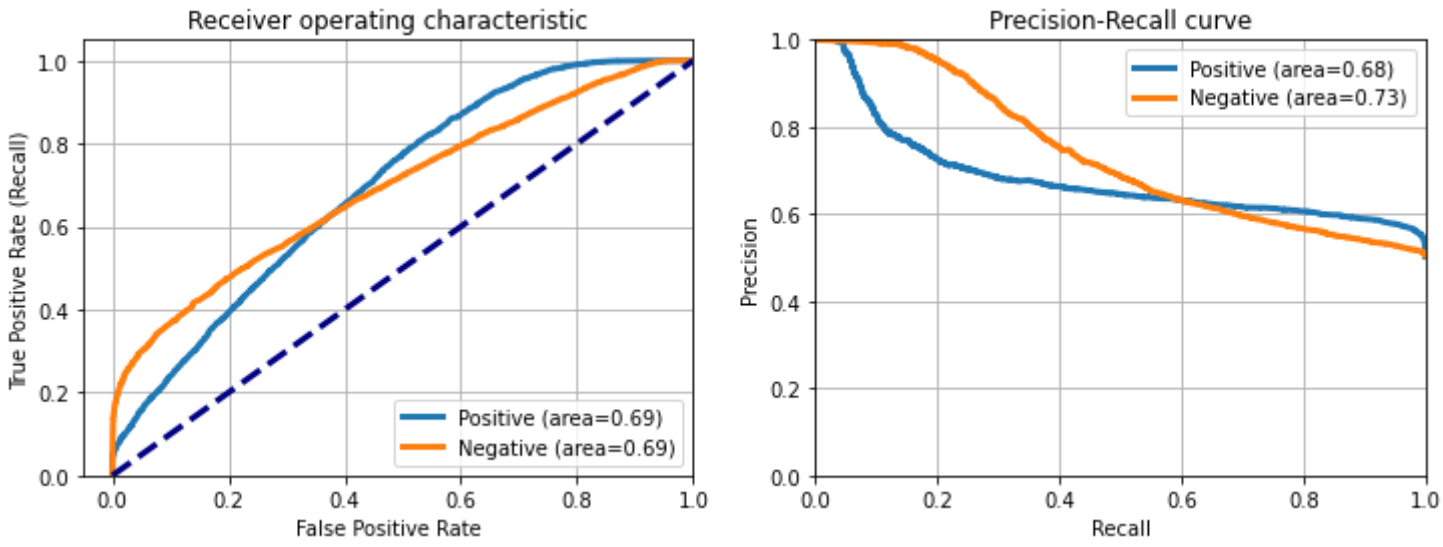
In [71]:
```
if bow_run:
# K-Neighbors classifier
    if neigbors:

        model = KNeighborsClassifier(n_neighbors=neighbors)
        calibrated = CalibratedClassifierCV(base_estimator=model,cv=k_fold)
        calibrated.fit(X_train, y_train)
        score = calibrated.score(X_test, y_test)
        result["KNeighborsClassifier"] = score
        print("Score for K-Neighbors classifier with neighbors={}, fold={} > {} %".format(neighbors, k_fold, score))

        if save:
            dump(calibrated, "saved_runs/kneighbors_calibrated.joblib")
```

```
Score for K-Neighbors classifier with neighbors=10, fold=5 > 0.6270291974989961 %
```

In [72]:
```
predicted = calibrated.predict(X_test)
predicted_prob = calibrated.predict_proba(X_test)

res1, res2 = map(list, zip(*predicted_prob))

accuracy = metrics.accuracy_score(y_test, predicted)
auc = metrics.roc_auc_score(y_test, res1, multi_class="ovr")
print("Accuracy:{:^20}".format(round(accuracy,2)))
print("Auc:{:^30}".format(round(auc,2)))
print("Detail:")
print(metrics.classification_report(y_test, predicted))

## Plot confusion matrix
cm = metrics.confusion_matrix(y_test, predicted)
fig, ax = plt.subplots()
sns.heatmap(cm, annot=True, fmt='d', ax=ax, cmap=plt.cm.Blues,
            cbar=False)
ax.set(xlabel="Pred", ylabel="True", xticklabels=classes,
       yticklabels=classes, title="Confusion matrix")
plt.yticks(rotation=0)
plt.gcf().set_size_inches(11.5, 4)

fig, ax = plt.subplots(nrows=1, ncols=2)
## Plot roc
for i in range(len(classes)):
    fpr, tpr, thresholds = metrics.roc_curve(y_test_array[:,i],  predicted_prob[:,i])
    ax[0].plot(fpr, tpr, lw=3,
label='{0} (area={1:0.2f})'.format(classes[i],
                                   metrics.auc(fpr, tpr))
                )
ax[0].plot([0,1], [0,1], color='navy', lw=3, linestyle='--')
ax[0].set(xlim=[-0.05,1.0], ylim=[0.0,1.05],
          xlabel='False Positive Rate',
          ylabel="True Positive Rate (Recall)",
          title="Receiver operating characteristic")
ax[0].legend(loc="lower right")
ax[0].grid(True)

## Plot precision-recall curve
for i in range(len(classes)):
    precision, recall, thresholds = metrics.precision_recall_curve(
                 y_test_array[:,i], predicted_prob[:,i])
    ax[1].plot(recall, precision, lw=3,
                label='{0} (area={1:0.2f})'.format(classes[i],
                                    metrics.auc(recall, precision))
               )
ax[1].set(xlim=[0.0,1.05], ylim=[0.0,1.05], xlabel='Recall',
          ylabel="Precision", title="Precision-Recall curve")
ax[1].legend(loc="best")
ax[1].grid(True)
plt.gcf().set_size_inches(12, 4)
plt.xlim(0, 1)
plt.ylim(0, 1)
print(end="")
```

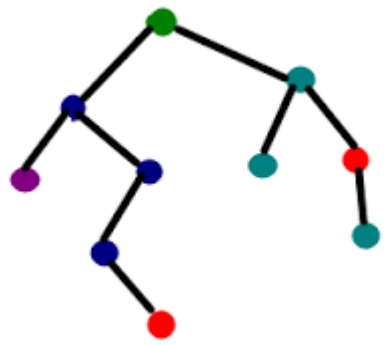
```
Accuracy:        0.63
Auc:             0.31
Detail:
              precision    recall  f1-score   support

           0       0.62      0.65      0.64      8755
           1       0.63      0.60      0.62      8678

    accuracy                           0.63     17433
   macro avg       0.63      0.63      0.63     17433
weighted avg       0.63      0.63      0.63     17433
```





Decision tree

Each node splits the decision and the more nodes we have, the more accurate the decision tree will be. The last nodes of the decision tree, the leaf, is where the decision is being made.

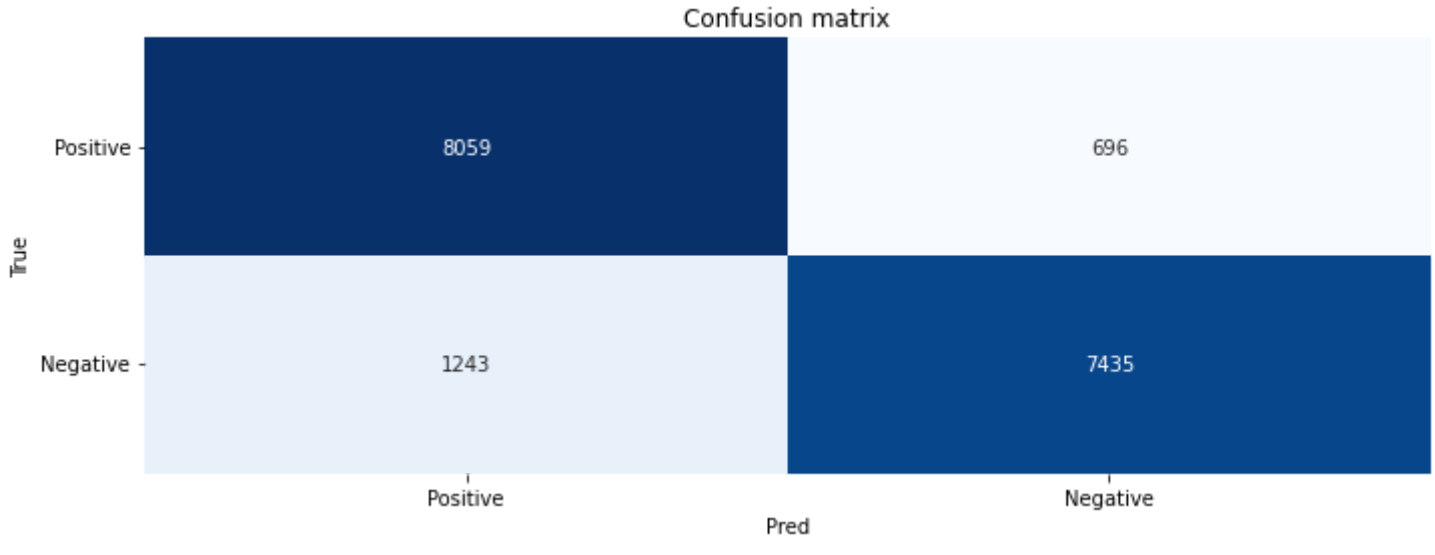


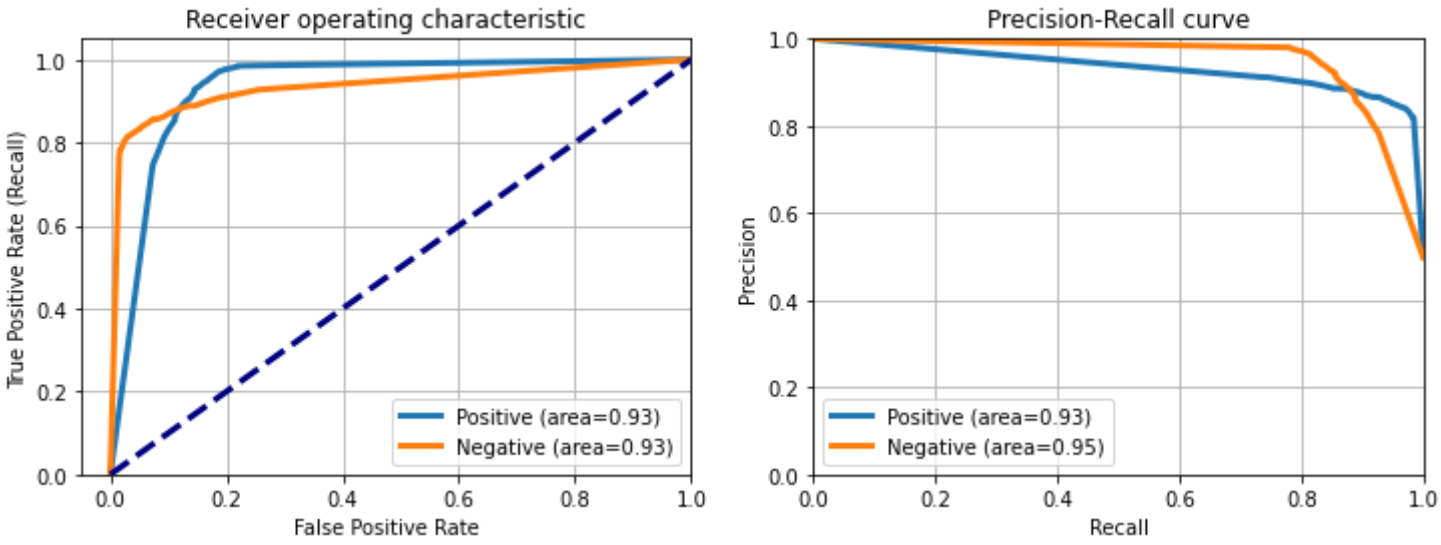
In [73]:
```
if bow_run:
# Decision tree classifier
    if tree:

        model = DecisionTreeClassifier(min_samples_split=min_samples_split, min_samples_leaf=min_samples_leaf)
        calibrated = CalibratedClassifierCV(base_estimator=model,cv=k_fold)
        calibrated.fit(X_train, y_train)
        score = calibrated.score(X_test, y_test)
        result["DecisionTreeClassifier"] = score
        print("Score for Decision tree classifier with min_samples_split={}, min_samples_leaf={}, fold={} > {} %".format

        if save:
            dump(calibrated, "saved_runs/tree_calibrated.joblib")
```

```
Score for Decision tree classifier with min_samples_split=2, min_samples_leaf=1, fold=5 > 0.8887741639419492 %
```

In [74]:
```
predicted = calibrated.predict(X_test)
predicted_prob = calibrated.predict_proba(X_test)

res1, res2 = map(list, zip(*predicted_prob))

accuracy = metrics.accuracy_score(y_test, predicted)
auc = metrics.roc_auc_score(y_test, res1, multi_class="ovr")
print("Accuracy:",  round(accuracy,2))
print("Auc:", round(auc,2))
print("Detail:")
print(metrics.classification_report(y_test, predicted))

## Plot confusion matrix
cm = metrics.confusion_matrix(y_test, predicted)
fig, ax = plt.subplots()
sns.heatmap(cm, annot=True, fmt='d', ax=ax, cmap=plt.cm.Blues,
            cbar=False)
ax.set(xlabel="Pred", ylabel="True", xticklabels=classes,
       yticklabels=classes, title="Confusion matrix")
plt.yticks(rotation=0)
plt.gcf().set_size_inches(11.5, 4)

fig, ax = plt.subplots(nrows=1, ncols=2)
## Plot roc
for i in range(len(classes)):
    fpr, tpr, thresholds = metrics.roc_curve(y_test_array[:,i],  predicted_prob[:,i])
    ax[0].plot(fpr, tpr, lw=3,
label='{0} (area={1:0.2f})'.format(classes[i],
                                   metrics.auc(fpr, tpr))
                )
ax[0].plot([0,1], [0,1], color='navy', lw=3, linestyle='--')
ax[0].set(xlim=[-0.05,1.0], ylim=[0.0,1.05],
          xlabel='False Positive Rate',
          ylabel="True Positive Rate (Recall)",
          title="Receiver operating characteristic")
ax[0].legend(loc="lower right")
ax[0].grid(True)

## Plot precision-recall curve
for i in range(len(classes)):
    precision, recall, thresholds = metrics.precision_recall_curve(
                 y_test_array[:,i], predicted_prob[:,i])
    ax[1].plot(recall, precision, lw=3,
               label='{0} (area={1:0.2f})'.format(classes[i],
                                  metrics.auc(recall, precision))
              )
ax[1].set(xlim=[0.0,1.05], ylim=[0.0,1.05], xlabel='Recall',
          ylabel="Precision", title="Precision-Recall curve")
ax[1].legend(loc="best")
ax[1].grid(True)
plt.gcf().set_size_inches(12, 4)
plt.xlim(0, 1)
plt.ylim(0, 1)
print(end="")
```

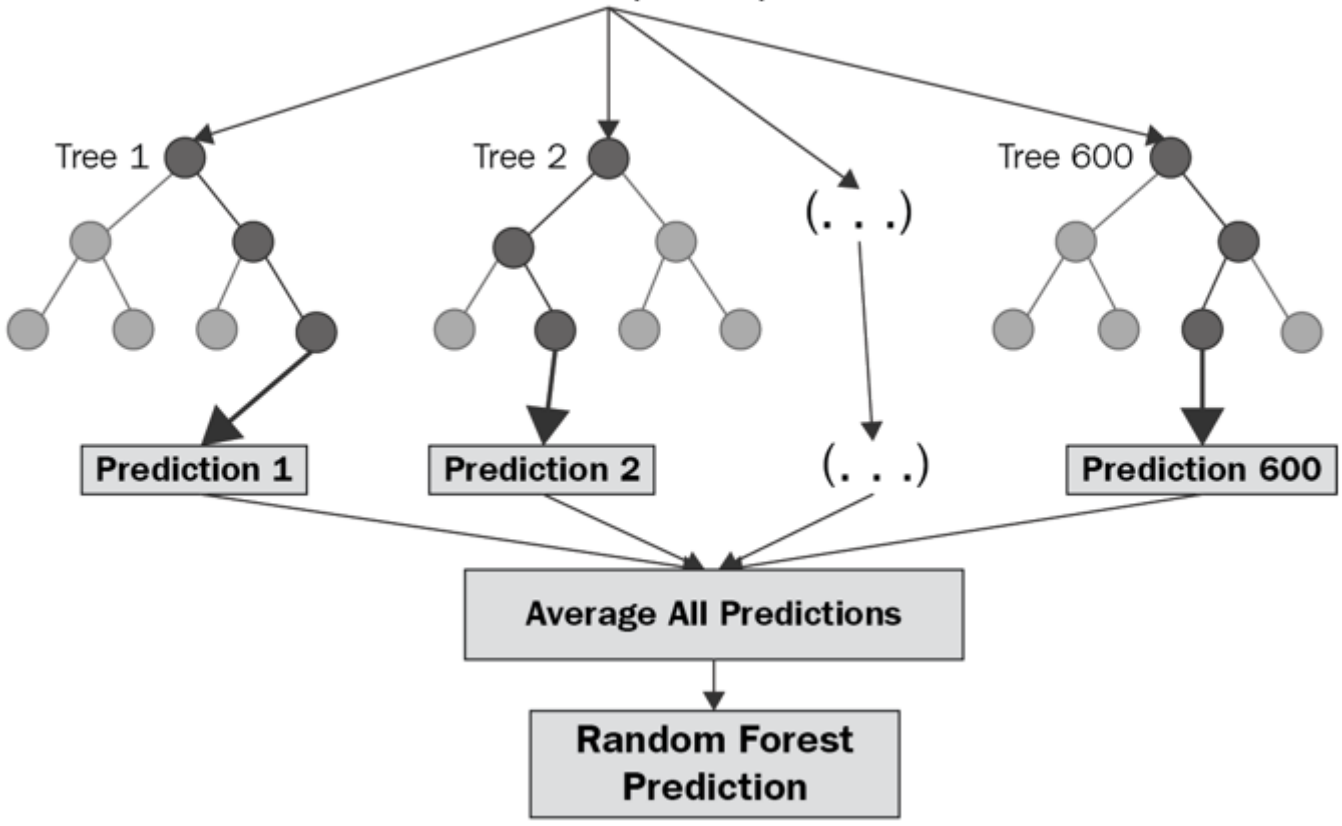
```
Accuracy: 0.89
Auc: 0.07
Detail:
              precision    recall  f1-score   support

           0       0.87      0.92      0.89      8755
           1       0.91      0.86      0.88      8678

    accuracy                           0.89     17433
   macro avg       0.89      0.89      0.89     17433
weighted avg       0.89      0.89      0.89     17433
```





RandomForestClassifier

Random forests are an ensemble learning technique that builds off of decision trees. Random forests involve creating multiple decision trees using bootstrapped datasets of the original data and randomly selecting a subset of variables at each step of the decision tree. The model then selects the mode of all of the predictions of each decision tree. What's the point of this? By relying on a "majority wins" model, it reduces the risk of error from an individual tree

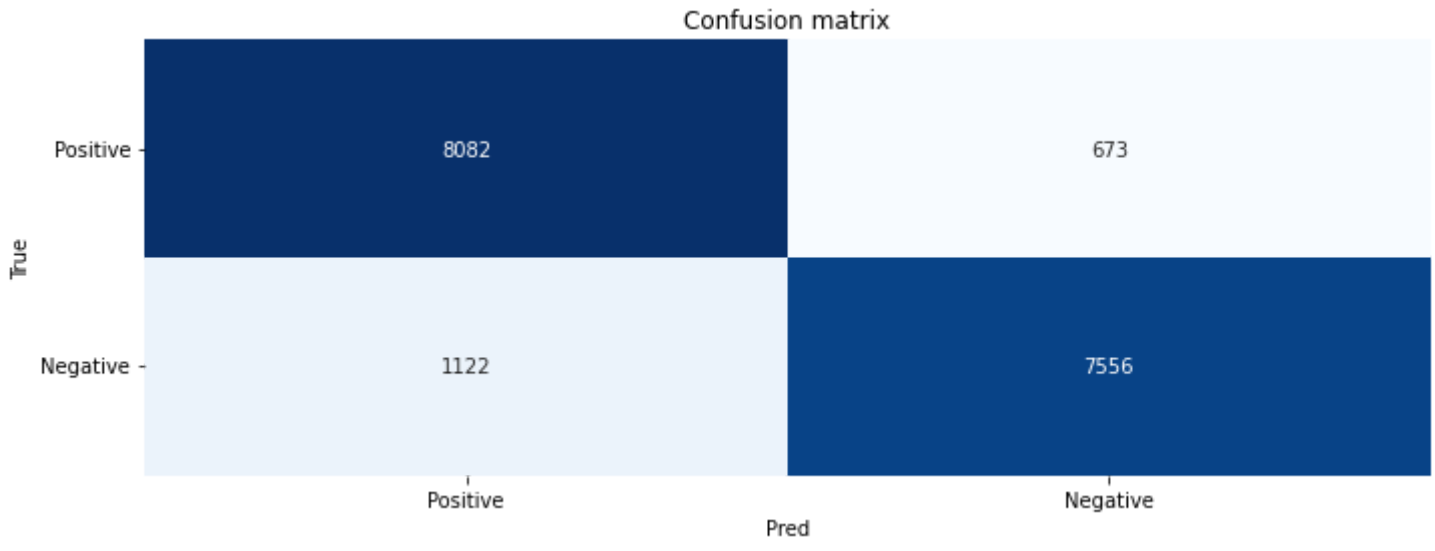


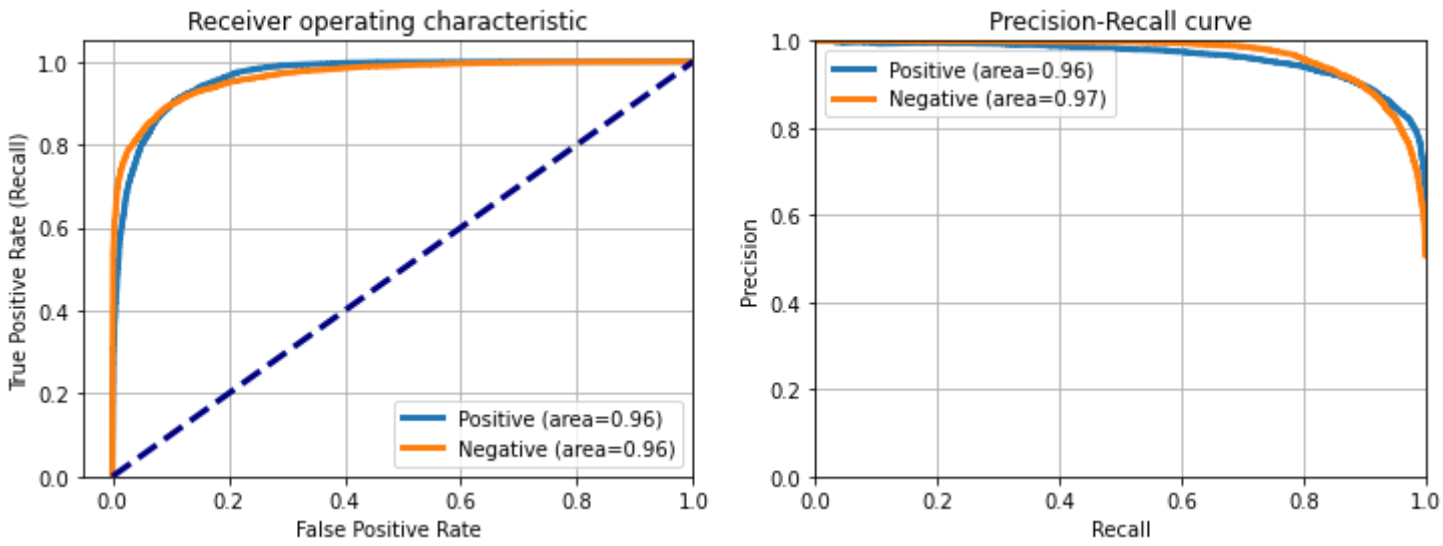
In [75]:
```
if bow_run:
# RandomForestClassifier
    if random_forest:
        model = RandomForestClassifier()
        calibrated = CalibratedClassifierCV(base_estimator=model,cv=k_fold)
        calibrated.fit(X_train, y_train)
        score = calibrated.score(X_test, y_test)
        result["LogisticRegression"] = score
        print("Score for RandomForestClassifier classifier with fold={} > {} %".format(k_fold, score))

        if save:
            dump(calibrated, "saved_runs/RandomForestClassifier_calibrated.joblib")
```

```
Score for RandomForestClassifier classifier with fold=5 > 0.8970343601216084 %
```

In [76]:
```
predicted = calibrated.predict(X_test)
predicted_prob = calibrated.predict_proba(X_test)

res1, res2 = map(list, zip(*predicted_prob))

accuracy = metrics.accuracy_score(y_test, predicted)
auc = metrics.roc_auc_score(y_test, res1, multi_class="ovr")
print("Accuracy:{:^20}".format(round(accuracy,2)))
print("Auc:{:^30}".format(round(auc,2)))
print("Detail:")
print(metrics.classification_report(y_test, predicted))

## Plot confusion matrix
cm = metrics.confusion_matrix(y_test, predicted)
fig, ax = plt.subplots()
sns.heatmap(cm, annot=True, fmt='d', ax=ax, cmap=plt.cm.Blues,
            cbar=False)
ax.set(xlabel="Pred", ylabel="True", xticklabels=classes,
       yticklabels=classes, title="Confusion matrix")
plt.yticks(rotation=0)
plt.gcf().set_size_inches(11.5, 4)

fig, ax = plt.subplots(nrows=1, ncols=2)
## Plot roc
for i in range(len(classes)):
    fpr, tpr, thresholds = metrics.roc_curve(y_test_array[:,i],  predicted_prob[:,i])
    ax[0].plot(fpr, tpr, lw=3,
label='{0} (area={1:0.2f})'.format(classes[i],
                                   metrics.auc(fpr, tpr))
                )
ax[0].plot([0,1], [0,1], color='navy', lw=3, linestyle='--')
ax[0].set(xlim=[-0.05,1.0], ylim=[0.0,1.05],
          xlabel='False Positive Rate',
          ylabel="True Positive Rate (Recall)",
          title="Receiver operating characteristic")
ax[0].legend(loc="lower right")
ax[0].grid(True)

## Plot precision-recall curve
for i in range(len(classes)):
    precision, recall, thresholds = metrics.precision_recall_curve(
                 y_test_array[:,i], predicted_prob[:,i])
    ax[1].plot(recall, precision, lw=3,
                label='{0} (area={1:0.2f})'.format(classes[i],
                                    metrics.auc(recall, precision))
               )
ax[1].set(xlim=[0.0,1.05], ylim=[0.0,1.05], xlabel='Recall',
          ylabel="Precision", title="Precision-Recall curve")
ax[1].legend(loc="best")
ax[1].grid(True)
plt.gcf().set_size_inches(12, 4)
plt.xlim(0, 1)
plt.ylim(0, 1)
print(end="")
```

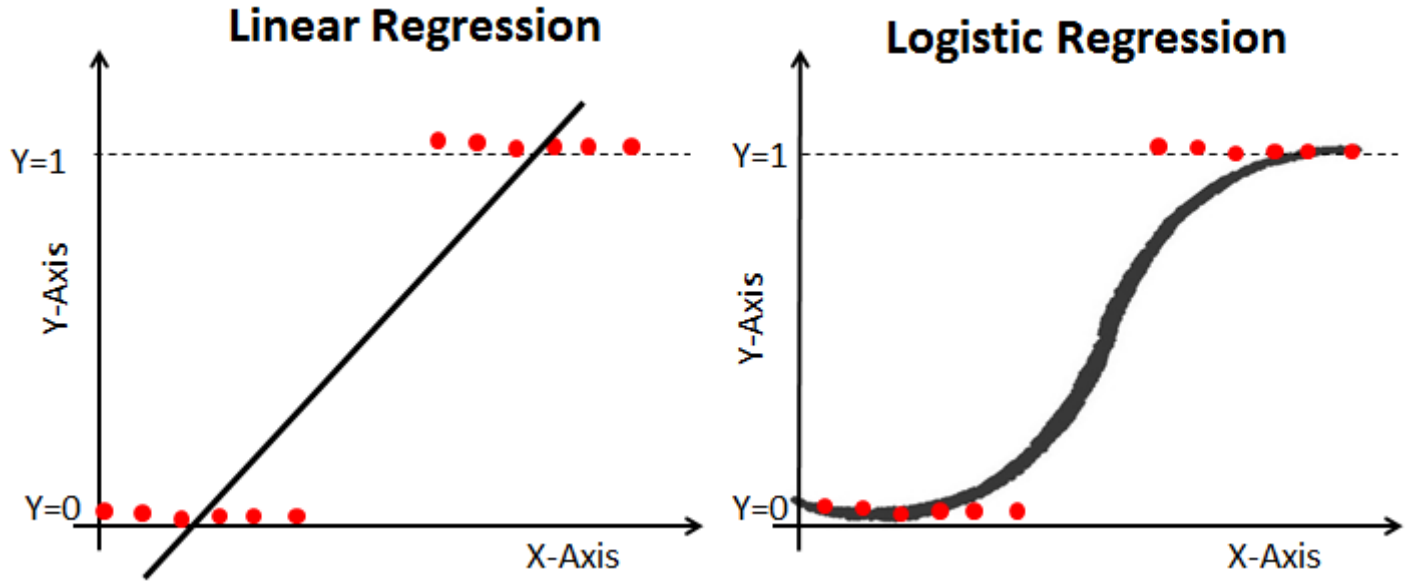
```
Accuracy:        0.9
Auc:             0.04
Detail:
              precision    recall  f1-score   support

           0       0.88      0.92      0.90      8755
           1       0.92      0.87      0.89      8678

    accuracy                           0.90     17433
   macro avg       0.90      0.90      0.90     17433
weighted avg       0.90      0.90      0.90     17433
```





Logistic regression

Logistic regression is similar to linear regression but is used to model the probability of a finite number of outcomes, typically two. There are a number of reasons why logistic regression is used over linear regression when modeling probabilities of outcomes (see here). In essence, a logistic equation is created in such a way that the output values can only be between 0 and 1

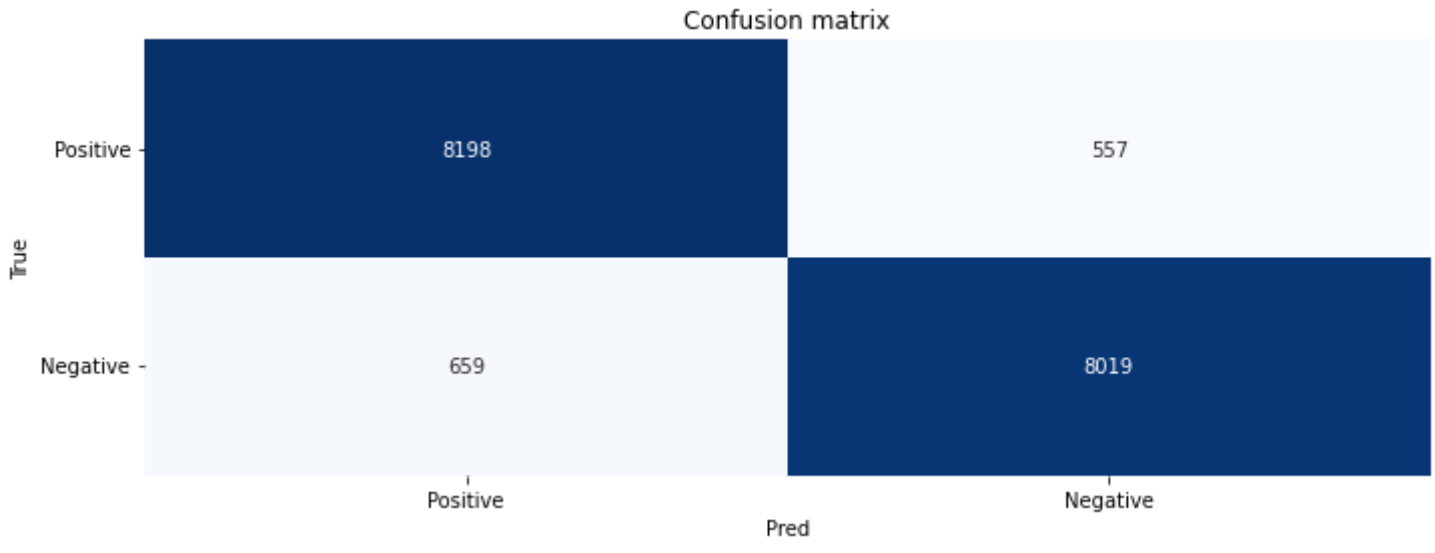


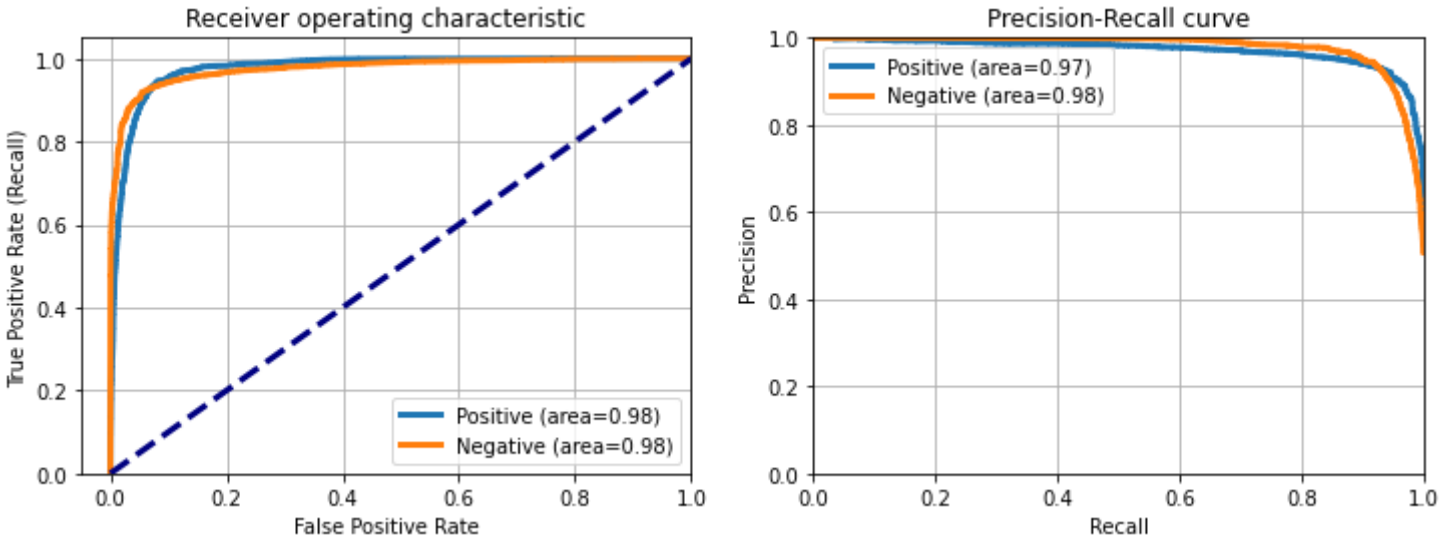
In [77]:
```
if bow_run:
# LogisticRegression
    if logistic:
        model = LogisticRegression()
        calibrated = CalibratedClassifierCV(base_estimator=model,cv=k_fold)
        calibrated.fit(X_train, y_train)
        score = calibrated.score(X_test, y_test)
        result["LogisticRegression"] = score
        print("Score for LogisticRegression classifier with fold={} > {} %".format(k_fold, score))

        if save:
            dump(calibrated, "saved_runs/logistic_reg_calibrated.joblib")
```

```
2021-10-04 18:50:12 [py.warnings] WARNING: C:\Users\vital\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qb
z5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.h
tml)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modu
les/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(

2021-10-04 18:50:14 [py.warnings] WARNING: C:\Users\vital\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qb
z5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.h
tml)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modu
les/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(

2021-10-04 18:50:17 [py.warnings] WARNING: C:\Users\vital\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qb
z5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.h
tml)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modu
les/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(


Score for LogisticRegression classifier with fold=5 > 0.9302472322606551 %
```

In [78]:
```
predicted = calibrated.predict(X_test)
predicted_prob = calibrated.predict_proba(X_test)

res1, res2 = map(list, zip(*predicted_prob))

accuracy = metrics.accuracy_score(y_test, predicted)
auc = metrics.roc_auc_score(y_test, res1, multi_class="ovr")
print("Accuracy:{:^20}".format(round(accuracy,2)))
print("Auc:{:^30}".format(round(auc,2)))
print("Detail:")
print(metrics.classification_report(y_test, predicted))

## Plot confusion matrix
cm = metrics.confusion_matrix(y_test, predicted)
fig, ax = plt.subplots()
sns.heatmap(cm, annot=True, fmt='d', ax=ax, cmap=plt.cm.Blues,
            cbar=False)
ax.set(xlabel="Pred", ylabel="True", xticklabels=classes,
       yticklabels=classes, title="Confusion matrix")
plt.yticks(rotation=0)
plt.gcf().set_size_inches(11.5, 4)

fig, ax = plt.subplots(nrows=1, ncols=2)
## Plot roc
for i in range(len(classes)):
    fpr, tpr, thresholds = metrics.roc_curve(y_test_array[:,i],  predicted_prob[:,i])
    ax[0].plot(fpr, tpr, lw=3,
label='{0} (area={1:0.2f})'.format(classes[i],
                                   metrics.auc(fpr, tpr))
                )
ax[0].plot([0,1], [0,1], color='navy', lw=3, linestyle='--')
ax[0].set(xlim=[-0.05,1.0], ylim=[0.0,1.05],
          xlabel='False Positive Rate',
          ylabel="True Positive Rate (Recall)",
          title="Receiver operating characteristic")
ax[0].legend(loc="lower right")
ax[0].grid(True)

## Plot precision-recall curve
for i in range(len(classes)):
    precision, recall, thresholds = metrics.precision_recall_curve(
                 y_test_array[:,i], predicted_prob[:,i])
    ax[1].plot(recall, precision, lw=3,
                label='{0} (area={1:0.2f})'.format(classes[i],
                                    metrics.auc(recall, precision))
               )
ax[1].set(xlim=[0.0,1.05], ylim=[0.0,1.05], xlabel='Recall',
          ylabel="Precision", title="Precision-Recall curve")
ax[1].legend(loc="best")
ax[1].grid(True)
plt.gcf().set_size_inches(12, 4)
plt.xlim(0, 1)
plt.ylim(0, 1)
print(end="")
```

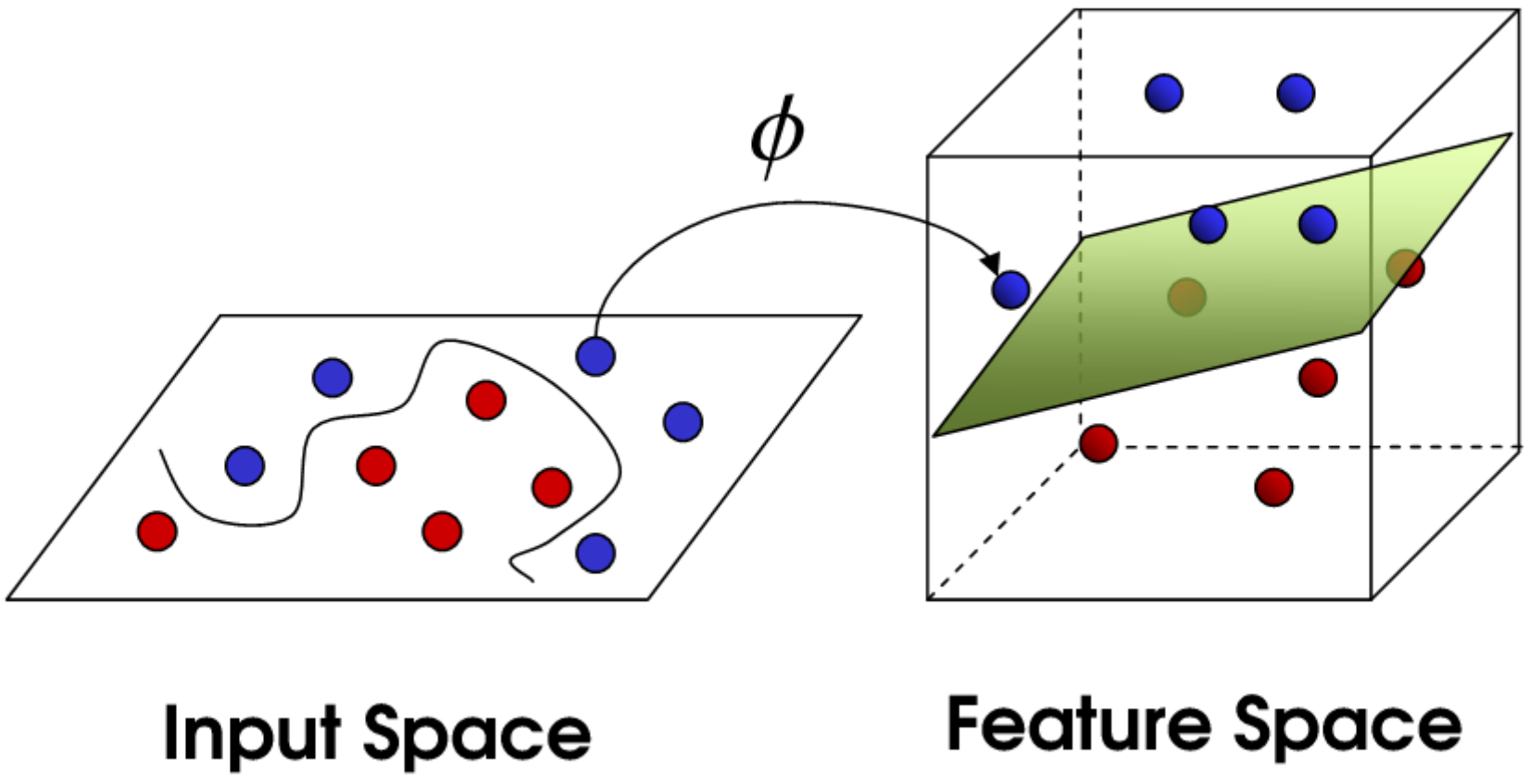
```
Accuracy:        0.93
Auc:             0.02
Detail:
              precision    recall  f1-score   support

           0       0.93      0.94      0.93      8755
           1       0.94      0.92      0.93      8678

    accuracy                           0.93     17433
   macro avg       0.93      0.93      0.93     17433
weighted avg       0.93      0.93      0.93     17433
```

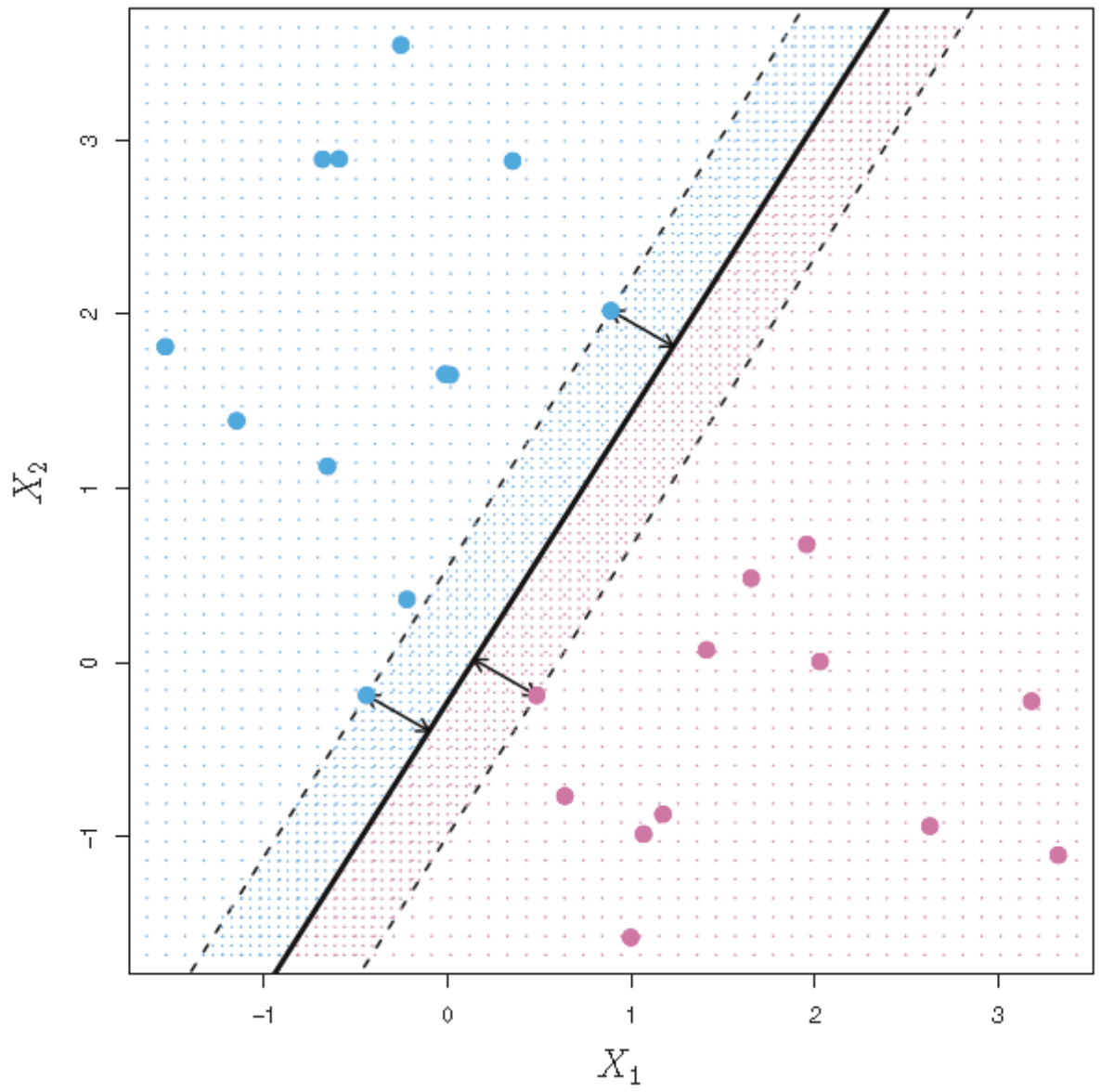




Support Vector Machine

SVM tries to finds the “best” margin (distance between the line and the support vectors) that separates the classes and this reduces the risk of error on the data. SVM works well with unstructured and semi-structured data like text and images while logistic regression works with already identified independent variables. The risk of overfitting is less in SVM, while Logistic regression is vulnerable to overfitting. The algorithm creates a hyperplane or line(decision boundary) which separates data into classes. It uses the kernel trick to find the best line separator (decision boundary that has same distance from the boundary point of both classes)



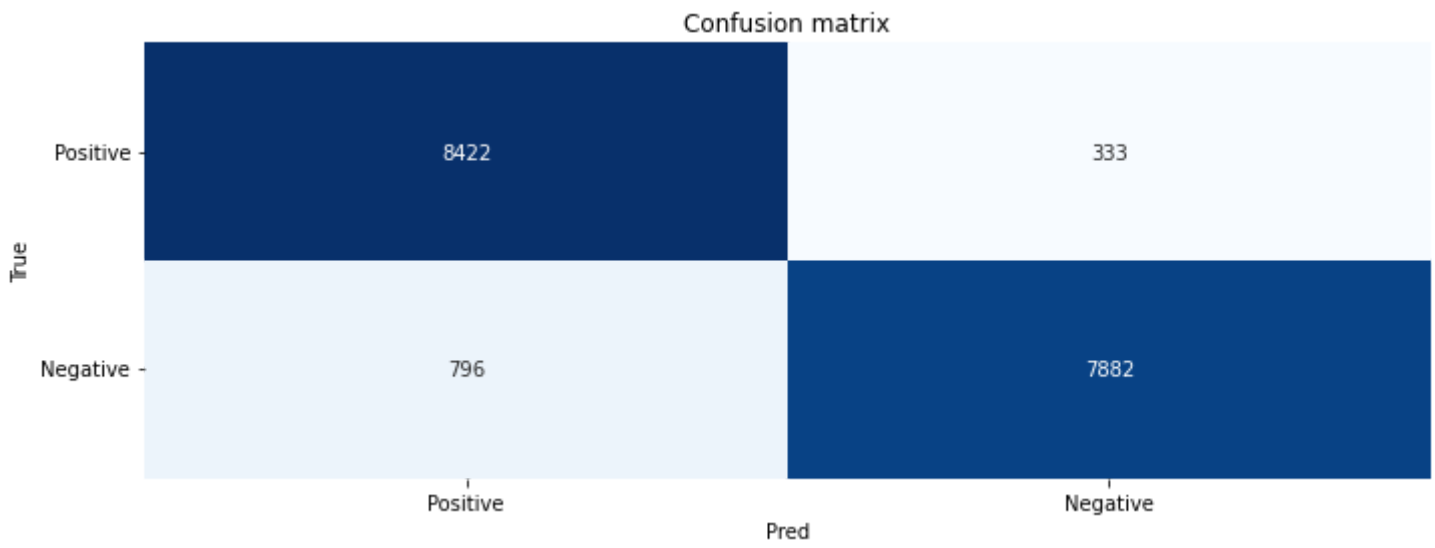


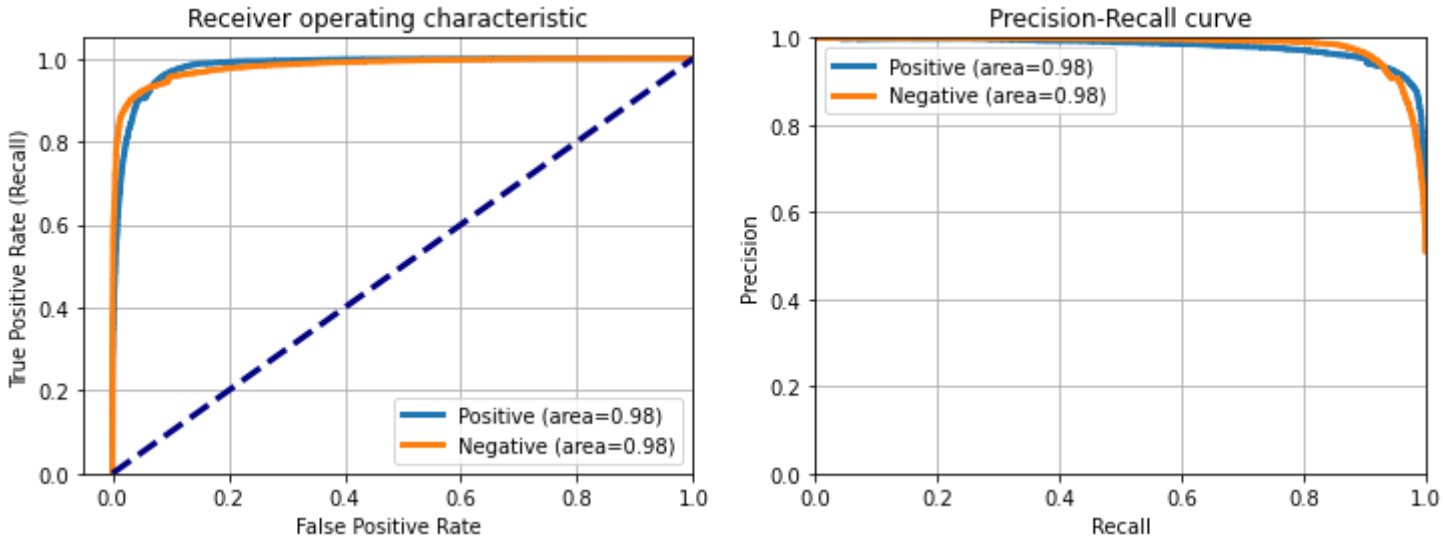
In [79]:
```
if bow_run:
# Linear support vector machine
    if svm:

        model = LinearSVC(dual=False, tol=tolerance, max_iter=max_iter_runtime)
        calibrated = CalibratedClassifierCV(base_estimator=model,cv=k_fold)
        calibrated.fit(X_train, y_train)
        score = calibrated.score(X_test, y_test)
        result["SVM"] = score
        print("Score for Linear support vector machine with fold={} = {} %".format(k_fold, score))
        # CalibratedClassifierCV -
        # This class uses cross-validation to both estimate the parameters of a classifier and subsequently calibrate a c
        # fits a copy of the base estimator to the training subset, and calibrates it using the testing subset

        if save:
            dump(calibrated, "saved_runs/linear_svc_calibrated.joblib")
```

```
Score for Linear support vector machine with fold=5 = 0.9352377674525325 %
```

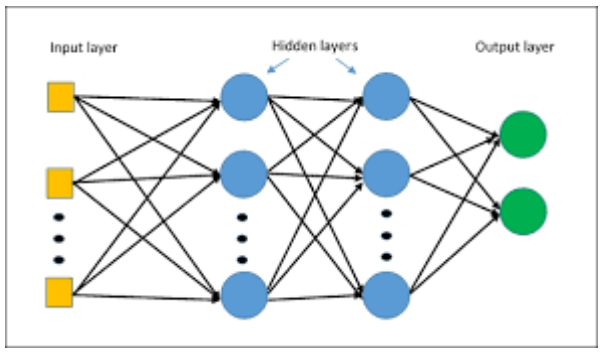
In [80]:
```
predicted = calibrated.predict(X_test)
predicted_prob = calibrated.predict_proba(X_test)

res1, res2 = map(list, zip(*predicted_prob))

accuracy = metrics.accuracy_score(y_test, predicted)
auc = metrics.roc_auc_score(y_test, res1, multi_class="ovr")
print("Accuracy:{:^20}".format(round(accuracy,2)))
print("Auc:{:^30}".format(round(auc,2)))
print("Detail:")
print(metrics.classification_report(y_test, predicted))

## Plot confusion matrix
cm = metrics.confusion_matrix(y_test, predicted)
fig, ax = plt.subplots()
sns.heatmap(cm, annot=True, fmt='d', ax=ax, cmap=plt.cm.Blues,
            cbar=False)
ax.set(xlabel="Pred", ylabel="True", xticklabels=classes,
       yticklabels=classes, title="Confusion matrix")
plt.yticks(rotation=0)
plt.gcf().set_size_inches(11.5, 4)

fig, ax = plt.subplots(nrows=1, ncols=2)
## Plot roc
for i in range(len(classes)):
    fpr, tpr, thresholds = metrics.roc_curve(y_test_array[:,i],  predicted_prob[:,i])
    ax[0].plot(fpr, tpr, lw=3,
label='{0} (area={1:0.2f})'.format(classes[i],
                                   metrics.auc(fpr, tpr))
                )
ax[0].plot([0,1], [0,1], color='navy', lw=3, linestyle='--')
ax[0].set(xlim=[-0.05,1.0], ylim=[0.0,1.05],
          xlabel='False Positive Rate',
          ylabel="True Positive Rate (Recall)",
          title="Receiver operating characteristic")
ax[0].legend(loc="lower right")
ax[0].grid(True)

## Plot precision-recall curve
for i in range(len(classes)):
    precision, recall, thresholds = metrics.precision_recall_curve(
                 y_test_array[:,i], predicted_prob[:,i])
    ax[1].plot(recall, precision, lw=3,
               label='{0} (area={1:0.2f})'.format(classes[i],
                                  metrics.auc(recall, precision))
              )
ax[1].set(xlim=[0.0,1.05], ylim=[0.0,1.05], xlabel='Recall',
          ylabel="Precision", title="Precision-Recall curve")
ax[1].legend(loc="best")
ax[1].grid(True)
plt.gcf().set_size_inches(12, 4)
plt.xlim(0, 1)
plt.ylim(0, 1)
print(end="")
```

```
Accuracy:        0.94
Auc:             0.02
Detail:
              precision    recall  f1-score   support

           0       0.91      0.96      0.94      8755
           1       0.96      0.91      0.93      8678

    accuracy                           0.94     17433
   macro avg       0.94      0.94      0.94     17433
weighted avg       0.94      0.94      0.94     17433
```





Keras Multilayer Perceptron forward network model with fully connected layers



In [ ]:

```
"""

Currently disabled due to RAM optimizations

if bow_run:
# neural network
    if neural:
        X_train = train["sentences"]
        y_train = train['Negative']
        X_test = test["sentences"]
        y_test = test['Negative']


        tokenizer = Tokenizer()
        tokenizer.fit_on_texts(balanced["sentences"])

        #Xtrain = tokenizer.texts_to_matrix(X_train, mode='binary') #  mode="binary" or "freq"
        #Xtest = tokenizer.texts_to_matrix(X_test, mode='binary')   #  mode="binary" or "freq"

        Xtrain = []
        for line in X_train:
            # Converting the lines into matrix, line-by-line.
            m = tokenizer.texts_to_matrix([line], mode='binary')[0]
            Xtrain.append(m)

        Xtest = []
        for line in X_test:
            # Converting the lines into matrix, line-by-line.
            m = tokenizer.texts_to_matrix([line], mode='binary')[0]
            Xtest.append(m)


        n_words = Xtest.shape[1]

        # define network
        model = Sequential()
        model.add(Dense(10, input_shape=(n_words,), activation='relu'))
        model.add(Dense(1, activation='sigmoid'))
        # compile network
        model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
        # fit network
        model.fit(Xtrain, ytrain, epochs=nb_epoch, verbose=2)
        # evaluate
        loss, acc = model.evaluate(Xtest, y_test, verbose=0)
        print("loss: {}, acc: {}".format(loss, acc))
        result["network"] = acc

        if save:
            dump(model, "saved_runs/network_calibrated.joblib")


"""
```

In [83]:
```
print("\tScore summary for bag of words\n")
for key in result:
    print("{:<22}: {:>22}".format(key, result[key]))

print("\n\n\t\t  Max value\n")
print("{:<22}: {:>22}".format(max(result), result[max(result)]))
```

```
        Score summary for bag of words

BernoulliNB           :     0.7640107841450123
ComplementNB          :      0.906384443297195
MultinomialNB         :      0.906384443297195
KNeighborsClassifier  :     0.6270291974989961
DecisionTreeClassifier:     0.8887741639419492
LogisticRegression    :     0.9302472322606551
SVM                   :     0.9352377674525325


                  Max value

SVM                   :     0.9352377674525325
```

Linear support vector machine recievs the best score with

Need to complete BERT model and ELMO models

Early Results

In [81]:
```
sample_text_a = ["You are shit"]
sample_text_b = ["Fuck"]
sample_text_c = ["I love you"]
sample_text_d = ["Idan is the best teach in the Open University"]
sample_text_e = ["I suck the water outside of the ship"]
sample_text_f = ["I drain the water outside of the ship"]
sample_text_g = ["The data I am using might or might not be refined to be better"]


list_of_samples = [sample_text_a, sample_text_b, sample_text_c, sample_text_d, sample_text_e, sample_text_f, sample_text

def _get_prob(prob):
  return prob[1]

def probability(texts):
  return np.apply_along_axis(_get_prob, 1, calibrated.predict_proba(bag_of_words.transform(texts)))


# Print the result per sample
for sample in list_of_samples:
    print(np.abs(1-calibrated.predict(bag_of_words.transform(sample))), end="")
    prob = probability(sample)
    if prob[0] >= 0.5:
        print("Negative {}%: {}".format(int(np.round(prob[0], 2)*100), sample))
    else:
        print("Positive {}%: {}".format(int(np.round(1-prob[0], 2)*100), sample))
```

```
[0]Negative 100%: ['You are shit']
[0]Negative 100%: ['Fuck']
[1]Positive 86%: ['I love you']
[1]Positive 90%: ['Idan is the best teach in the Open University']
[0]Negative 89%: ['I suck the water outside of the ship']
[1]Positive 80%: ['I drain the water outside of the ship']
[1]Positive 82%: ['The data I am using might or might not be refined to be better']
```

It can be seen that "I suck the water outside of the ship" failed

Additional features to be created for the final project

Image to text - recognize samples so we can later use our model to evaluate them

In [82]:
```
# Sample from "http://digitalnativestudios.com/textmeshpro/docs/rich-text/"
source_path = r"materials\images\in\image_text_sample.png"

image = cv2.imread(source_path)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Remove some noises, this should be later changed to be dynamic
gray = cv2.medianBlur(gray, 3)

source_path_out = "\\".join(source_path.split("\\")[0:-2])
filename = source_path_out + "\\out\\" + "image_out.png"
cv2.imwrite(filename, gray)

# The line below runs pytesseract, changing the stdout to text file that can be later loaded to the code above
# text = pytesseract.image_to_string(Image.open(filename))
```

Out[82]: True

Google OCR (https://github.com/tesseract-ocr/tesseract)

Voice to text - recognize samples so we can later use our model to evaluate them

Under development

Found a cool open-source library to help split the audio into freq bins (fft) and a dataset that helps analyze feelings according the the sound (frequences, rate of change and etc)

librosa - OpenSource (https://librosa.org/doc/latest/index.html)

Data set

To do list

1. Add Bert and ELMO
2. Finialize the features
3. Clean the data better (preprocessing for the data)
4. Make a conclusion and "in a nutshell segments"
5. Finilze notebook look

In [ ]: