Ekstrakcja informacji

March 13, 2022

[96]:
```
import random
import plotly.express as px
import numpy as np
import pandas as pd
import nltk
```

https://github.com/sdadas/polish-nlp-resources

[97]:
```
ps = nltk.stem.PorterStemmer()

for w in ["program", "programs", "programmer", "programming", "programmers"]:
    print(w, " : ", ps.stem(w))
```

```
program  :  program
programs  :  program
programmer  :  programm
programming  :  program
programmers  :  programm
```

[98]:
```
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /home/maciej/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/maciej/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[98]:
```
True
```

[99]:
```
text = """Python is dynamically-typed and garbage-collected. It supports␣
 ↪multiple programming paradigms, including structured (particularly,␣
 ↪procedural), object-oriented and functional programming. It is often␣
 ↪described as a "batteries included" language due to its comprehensive␣
 ↪standard library."""
nltk.tokenize.word_tokenize(text)
```

[99]:
```
['Python',
 'is',
```

```
 'dynamically-typed',
 'and',
 'garbage-collected',
 '.',
 'It',
 'supports',
 'multiple',
 'programming',
 'paradigms',
 ',',
 'including',
 'structured',
 '(',
 'particularly',
 ',',
 'procedural',
 ')',
 ',',
 'object-oriented',
 'and',
 'functional',
 'programming',
 '.',
 'It',
 'is',
 'often',
 'described',
 'as',
 'a',
 '``',
 'batteries',
 'included',
 "''",
 'language',
 'due',
 'to',
 'its',
 'comprehensive',
 'standard',
 'library',
 '.']
```

[100]:
```
nltk.tokenize.sent_tokenize(text)
```

[100]:
```
['Python is dynamically-typed and garbage-collected.',
 'It supports multiple programming paradigms, including structured
(particularly, procedural), object-oriented and functional programming.',
```

```
 'It is often described as a "batteries included" language due to its
comprehensive standard library.']
```

[101]:
```
nltk.corpus.stopwords.words('german')
```

[101]:
```
['aber',
 'alle',
 'allem',
 'allen',
 'aller',
 'alles',
 'als',
 'also',
 'am',
 'an',
 'ander',
 'andere',
 'anderem',
 'anderen',
 'anderer',
 'anderes',
 'anderm',
 'andern',
 'anderr',
 'anders',
 'auch',
 'auf',
 'aus',
 'bei',
 'bin',
 'bis',
 'bist',
 'da',
 'damit',
 'dann',
 'der',
 'den',
 'des',
 'dem',
 'die',
 'das',
 'dass',
 'daß',
 'derselbe',
 'derselben',
 'denselben',
 'desselben',
```

```
'demselben',
'dieselbe',
'dieselben',
'dasselbe',
'dazu',
'dein',
'deine',
'deinem',
'deinen',
'deiner',
'deines',
'denn',
'derer',
'dessen',
'dich',
'dir',
'du',
'dies',
'diese',
'diesem',
'diesen',
'dieser',
'dieses',
'doch',
'dort',
'durch',
'ein',
'eine',
'einem',
'einen',
'einer',
'eines',
'einig',
'einige',
'einigem',
'einigen',
'einiger',
'einiges',
'einmal',
'er',
'ihn',
'ihm',
'es',
'etwas',
'euer',
'eure',
'eurem',
```

```
'euren',
'eurer',
'eures',
'für',
'gegen',
'gewesen',
'hab',
'habe',
'haben',
'hat',
'hatte',
'hatten',
'hier',
'hin',
'hinter',
'ich',
'mich',
'mir',
'ihr',
'ihre',
'ihrem',
'ihren',
'ihrer',
'ihres',
'euch',
'im',
'in',
'indem',
'ins',
'ist',
'jede',
'jedem',
'jeden',
'jeder',
'jedes',
'jene',
'jenem',
'jenen',
'jener',
'jenes',
'jetzt',
'kann',
'kein',
'keine',
'keinem',
'keinen',
'keiner',
```

```
'keines',
'können',
'könnte',
'machen',
'man',
'manche',
'manchem',
'manchen',
'mancher',
'manches',
'mein',
'meine',
'meinem',
'meinen',
'meiner',
'meines',
'mit',
'muss',
'musste',
'nach',
'nicht',
'nichts',
'noch',
'nun',
'nur',
'ob',
'oder',
'ohne',
'sehr',
'sein',
'seine',
'seinem',
'seinen',
'seiner',
'seines',
'selbst',
'sich',
'sie',
'ihnen',
'sind',
'so',
'solche',
'solchem',
'solchen',
'solcher',
'solches',
'soll',
```

```
'sollte',
'sondern',
'sonst',
'über',
'um',
'und',
'uns',
'unsere',
'unserem',
'unseren',
'unser',
'unseres',
'unter',
'viel',
'vom',
'von',
'vor',
'während',
'war',
'waren',
'warst',
'was',
'weg',
'weil',
'weiter',
'welche',
'welchem',
'welchen',
'welcher',
'welches',
'wenn',
'werde',
'werden',
'wie',
'wieder',
'will',
'wir',
'wird',
'wirst',
'wo',
'wollen',
'wollte',
'würde',
'würden',
'zu',
'zum',
'zur',
```

```
 'zwar',
 'zwischen']
```

[102]:
```
nltk_tokens = nltk.word_tokenize(text)
print(list(nltk.bigrams(nltk_tokens)))
```

```
[('Python', 'is'), ('is', 'dynamically-typed'), ('dynamically-typed', 'and'),
('and', 'garbage-collected'), ('garbage-collected', '.'), ('.', 'It'), ('It',
'supports'), ('supports', 'multiple'), ('multiple', 'programming'),
('programming', 'paradigms'), ('paradigms', ','), (',', 'including'),
('including', 'structured'), ('structured', '('), ('(', 'particularly'),
('particularly', ','), (',', 'procedural'), ('procedural', ')'), (')', ','),
(',', 'object-oriented'), ('object-oriented', 'and'), ('and', 'functional'),
('functional', 'programming'), ('programming', '.'), ('.', 'It'), ('It', 'is'),
('is', 'often'), ('often', 'described'), ('described', 'as'), ('as', 'a'), ('a',
'```'), ('```', 'batteries'), ('batteries', 'included'), ('included', "'''"),
("'''", 'language'), ('language', 'due'), ('due', 'to'), ('to', 'its'), ('its',
'comprehensive'), ('comprehensive', 'standard'), ('standard', 'library'),
('library', '.')]
```

[103]:
```
df = pd.DataFrame([['ma', 20], ['ala', 15], ['psa', 10], ['kota', 10]],␣
 ↪columns=['słowo', 'liczba'])
fig = px.bar(df, x="słowo", y="liczba")
fig.show()
```

[104]:
```
df = pd.DataFrame([[random.choice(['ang','polski','hiszp']), np.random.
 ↪geometric(0.2)]  for i in range(5000) ], columns=['jezyk', 'dlugosc'])
fig = px.histogram(df, x="dlugosc",facet_row='jezyk',nbins=50, hover_data=df.
 ↪columns)
fig.show()
```

[105]:
```
?px.histogram
```

```
Signature:
px.histogram(

data_frame=None,
    x=None,
    y=None,
    color=None,

pattern_shape=None,

facet_row=None,

facet_col=None,
```

```
facet_col_wrap=0,

facet_row_spacing=None,

facet_col_spacing=None,

hover_name=None,

hover_data=None,

animation_frame=None,

animation_group=None,

category_orders=None,
    labels=None,
    color_discrete_sequence=None

,

color_discrete_map=None,

pattern_shape_sequence=None,

pattern_shape_map=None,

marginal=None,
    opacity=None,

orientation=None,

barmode='relative',
    barnorm=None,

histnorm=None,
    log_x=False,
    log_y=False,
    range_x=None,
    range_y=None,

histfunc=None,

cumulative=None,
    nbins=None,

text_auto=False,
    title=None,
```

```
template=None,
    width=None,
    height=None,
)
Docstring:
    In a histogram, rows of `data_frame` are grouped together into a
    rectangular mark to visualize the 1D distribution of an aggregate
    function `histfunc` (e.g. the count or sum) of the value `y` (or `x` if
    `orientation` is `'h'`).

Parameters
----------
data_frame: DataFrame or array-like or dict
    This argument needs to be passed for column names (and not keyword
    names) to be used. Array-like and dict are tranformed internally to a
    pandas DataFrame. Optional: if missing, a DataFrame gets constructed
    under the hood using the other arguments.
x: str or int or Series or array-like
    Either a name of a column in `data_frame`, or a pandas Series or
    array_like object. Values from this column or array_like are used to
    position marks along the x axis in cartesian coordinates. If
    `orientation` is `'h'`, these values are used as inputs to `histfunc`.
    Either `x` or `y` can optionally be a list of column references or
    array_likes,  in which case the data will be treated as if it were
    'wide' rather than 'long'.
y: str or int or Series or array-like
    Either a name of a column in `data_frame`, or a pandas Series or
    array_like object. Values from this column or array_like are used to
    position marks along the y axis in cartesian coordinates. If
    `orientation` is `'v'`, these values are used as inputs to `histfunc`.
    Either `x` or `y` can optionally be a list of column references or
    array_likes,  in which case the data will be treated as if it were
    'wide' rather than 'long'.
color: str or int or Series or array-like
    Either a name of a column in `data_frame`, or a pandas Series or
    array_like object. Values from this column or array_like are used to
    assign color to marks.
pattern_shape: str or int or Series or array-like
    Either a name of a column in `data_frame`, or a pandas Series or
    array_like object. Values from this column or array_like are used to
    assign pattern shapes to marks.
facet_row: str or int or Series or array-like
    Either a name of a column in `data_frame`, or a pandas Series or
    array_like object. Values from this column or array_like are used to
    assign marks to facetted subplots in the vertical direction.
facet_col: str or int or Series or array-like
    Either a name of a column in `data_frame`, or a pandas Series or
    array_like object. Values from this column or array_like are used to
```

```
    assign marks to facetted subplots in the horizontal direction.
facet_col_wrap: int
    Maximum number of facet columns. Wraps the column variable at this
    width, so that the column facets span multiple rows. Ignored if 0, and
    forced to 0 if `facet_row` or a `marginal` is set.
facet_row_spacing: float between 0 and 1
    Spacing between facet rows, in paper units. Default is 0.03 or 0.0.7
    when facet_col_wrap is used.
facet_col_spacing: float between 0 and 1
    Spacing between facet columns, in paper units Default is 0.02.
hover_name: str or int or Series or array-like
    Either a name of a column in `data_frame`, or a pandas Series or
    array_like object. Values from this column or array_like appear in bold
    in the hover tooltip.
hover_data: list of str or int, or Series or array-like, or dict
    Either a list of names of columns in `data_frame`, or pandas Series, or
    array_like objects or a dict with column names as keys, with values
    True (for default formatting) False (in order to remove this column
    from hover information), or a formatting string, for example ':.3f' or
    '|%a' or list-like data to appear in the hover tooltip or tuples with a
    bool or formatting string as first element, and list-like data to
    appear in hover as second element Values from these columns appear as
    extra data in the hover tooltip.
animation_frame: str or int or Series or array-like
    Either a name of a column in `data_frame`, or a pandas Series or
    array_like object. Values from this column or array_like are used to
    assign marks to animation frames.
animation_group: str or int or Series or array-like
    Either a name of a column in `data_frame`, or a pandas Series or
    array_like object. Values from this column or array_like are used to
    provide object-constancy across animation frames: rows with matching
    `animation_group`s will be treated as if they describe the same object
    in each frame.
category_orders: dict with str keys and list of str values (default `{}`)
    By default, in Python 3.6+, the order of categorical values in axes,
    legends and facets depends on the order in which these values are first
    encountered in `data_frame` (and no order is guaranteed by default in
    Python below 3.6). This parameter is used to force a specific ordering
    of values per column. The keys of this dict should correspond to column
    names, and the values should be lists of strings corresponding to the
    specific display order desired.
labels: dict with str keys and str values (default `{}`)
    By default, column names are used in the figure for axis titles, legend
    entries and hovers. This parameter allows this to be overridden. The
    keys of this dict should correspond to column names, and the values
    should correspond to the desired label to be displayed.
color_discrete_sequence: list of str
    Strings should define valid CSS-colors. When `color` is set and the
```

```
    values in the corresponding column are not numeric, values in that
    column are assigned colors by cycling through `color_discrete_sequence`
    in the order described in `category_orders`, unless the value of
    `color` is a key in `color_discrete_map`. Various useful color
    sequences are available in the `plotly.express.colors` submodules,
    specifically `plotly.express.colors.qualitative`.
color_discrete_map: dict with str keys and str values (default `{}`)
    String values should define valid CSS-colors Used to override
    `color_discrete_sequence` to assign a specific colors to marks
    corresponding with specific values. Keys in `color_discrete_map` should
    be values in the column denoted by `color`. Alternatively, if the
    values of `color` are valid colors, the string `'identity'` may be
    passed to cause them to be used directly.
pattern_shape_sequence: list of str
    Strings should define valid plotly.js patterns-shapes. When
    `pattern_shape` is set, values in that column are assigned patterns-
    shapes by cycling through `pattern_shape_sequence` in the order
    described in `category_orders`, unless the value of `pattern_shape` is
    a key in `pattern_shape_map`.
pattern_shape_map: dict with str keys and str values (default `{}`)
    Strings values define plotly.js patterns-shapes. Used to override
    `pattern_shape_sequences` to assign a specific patterns-shapes to lines
    corresponding with specific values. Keys in `pattern_shape_map` should
    be values in the column denoted by `pattern_shape`. Alternatively, if
    the values of `pattern_shape` are valid patterns-shapes names, the
    string `'identity'` may be passed to cause them to be used directly.
marginal: str
    One of `'rug'`, `'box'`, `'violin'`, or `'histogram'`. If set, a
    subplot is drawn alongside the main plot, visualizing the distribution.
opacity: float
    Value between 0 and 1. Sets the opacity for markers.
orientation: str, one of `'h'` for horizontal or `'v'` for vertical.
    (default `'v'` if `x` and `y` are provided and both continous or both
    categorical,  otherwise `'v'`(`'h'`) if `x`(`y`) is categorical and
    `y`(`x`) is continuous,  otherwise `'v'`(`'h'`) if only `x`(`y`) is
    provided)
barmode: str (default `'relative'`)
    One of `'group'`, `'overlay'` or `'relative'` In `'relative'` mode,
    bars are stacked above zero for positive values and below zero for
    negative values. In `'overlay'` mode, bars are drawn on top of one
    another. In `'group'` mode, bars are placed beside each other.
barnorm: str (default `None`)
    One of `'fraction'` or `'percent'`. If `'fraction'`, the value of each
    bar is divided by the sum of all values at that location coordinate.
    `'percent'` is the same but multiplied by 100 to show percentages.
    `None` will stack up all values at each location coordinate.
histnorm: str (default `None`)
    One of `'percent'`, `'probability'`, `'density'`, or `'probability
```

```
    density'` If `None`, the output of `histfunc` is used as is. If
    `'probability'`, the output of `histfunc` for a given bin is divided by
    the sum of the output of `histfunc` for all bins. If `'percent'`, the
    output of `histfunc` for a given bin is divided by the sum of the
    output of `histfunc` for all bins and multiplied by 100. If
    `'density'`, the output of `histfunc` for a given bin is divided by the
    size of the bin. If `'probability density'`, the output of `histfunc`
    for a given bin is normalized such that it corresponds to the
    probability that a random event whose distribution is described by the
    output of `histfunc` will fall into that bin.
log_x: boolean (default `False`)
    If `True`, the x-axis is log-scaled in cartesian coordinates.
log_y: boolean (default `False`)
    If `True`, the y-axis is log-scaled in cartesian coordinates.
range_x: list of two numbers
    If provided, overrides auto-scaling on the x-axis in cartesian
    coordinates.
range_y: list of two numbers
    If provided, overrides auto-scaling on the y-axis in cartesian
    coordinates.
histfunc: str (default `'count'` if no arguments are provided, else `'sum'`)
    One of `'count'`, `'sum'`, `'avg'`, `'min'`, or `'max'`.Function used
    to aggregate values for summarization (note: can be normalized with
    `histnorm`). The arguments to this function are the values of `y`(`x`)
    if `orientation` is `'v'`(`'h'`).
cumulative: boolean (default `False`)
    If `True`, histogram values are cumulative.
nbins: int
    Positive integer. Sets the number of bins.
text_auto: bool or string (default `False`)
    If `True` or a string, the x or y or z values will be displayed as
    text, depending on the orientation A string like `'.2f'` will be
    interpreted as a `texttemplate` numeric formatting directive.
title: str
    The figure title.
template: str or dict or plotly.graph_objects.layout.Template instance
    The figure template name (must be a key in plotly.io.templates) or
    definition.
width: int (default `None`)
    The figure width in pixels.
height: int (default `None`)
    The figure height in pixels.

Returns
-------
    plotly.graph_objects.Figure
File:      ~/.local/lib/python3.8/site-
packages/plotly/express/_chart_types.py
```

Type: function

0.1 ZADANIE 1

(20 punktów)

ZNAJDŹ PRZYKŁAD TEKSTÓW Z TEJ SAMEJ DOMENY 1_000_000 słów albo nawet tłumaczenie : - język angielski - język polski - język z rodziny romańskich

Proponowane narzędzia: - nltk - plotly express - biblioteka collections - spacy (niekoniecznie)

Dla każdego z języków: - policz ilosć unikalnych lowercase słów (ze stemmingiem i bez) - policz ilosć znaków - policz ilosć unikalnych znaków - policz ilosć zdań zdań - policz ilosć unikalnych zdań - podaj min, max, średnią oraz medianę ilości znaków w słowie - podaj min, max, średnią oraz medianę ilości słów w zdaniu, znajdz najkrotsze i najdluzsze zdania - wygeneruj word cloud (normalnie i po usunięciu stopwordów) - wypisz 20 najbardziej popularnych słów (normalnie i po usunięciu stopwordów) (lowercase) - wypisz 20 najbardziej popularnych bigramów (normalnie i po usunięciu stopwordów) - narysuj wykres częstotliwości słów (histogram lub linie) w taki sposób żeby był czytelny, wypróbuj skali logarytmicznej dla osi x (ale na razie nie dla y), usuwanie słów poniżej limitu wystąpień itp. - punkt jak wyżej, tylko dla bigramów - punkt jak wyżej, tylko dla znaków - narysuj wykres barplot dla części mowy (PART OF SPEECH TAGS, tylko pierwszy stopień zagłębienia) - dla próbki 10000 zdań sprawdź jak często langdetect https://pypi.org/project/langdetect/ się myli i w jaki sposób. - zilustruj prawo zipfa (px.line z zaznaczonymi punktami) - napisz wnioski (10-50 zdań)

START ZADANIA

[106]:
```
!mkdir -p outputs

import requests
from bs4 import BeautifulSoup
txts = [
    ['pl', 'https://www.odaha.com/littleprince.php?f=MalyKsiaze'],
    ['en', 'https://www.odaha.com/antoine-de-saint-exupery/maly-princ/
 ↪the-little-prince'],
    ['it',  'https://www.odaha.com/littleprince.php?f=Italiano']
]

for k, v in txts:
    soup = BeautifulSoup(requests.get(v).content, "html.parser")
    t = soup.find_all("div", {"class": "entrytext"})
    txt = t[0].get_text()
    with open(f"outputs//prince_{k}.txt", "w",  encoding="utf-8") as f:
        f.write(txt)
```

[107]:
```
LANGUAGES = {"en": "Angielski", "pl": "Polski", "it": "Włoski"}
TXT = {}
for l in LANGUAGES:
    TXT[l] = open(f"outputs//prince_{l}.txt", "r").read()
    print(f'{LANGUAGES[l]} length {len(TXT[l])}')
```

```
Angielski length 90208
Polski length 75013
Włoski length 74078
```

- policz ilosć unikalnych lowercase słów (ze stemmingiem i bez)

[108]:
```python
import regex

def lowercase(text):
  for match in regex.finditer(r'[\p{L}0-9\*]+', text):
     yield match.group(0)

for l in LANGUAGES:
  lower = [word for word in lowercase(TXT[l]) if word[0].lower() == word[0]]
  unique_lower = set(lower)
  print(f"Unikalne {LANGUAGES[l]} (bez steamingu): {len(unique_lower)}")

  ps = nltk.stem.PorterStemmer()
  lower_stemmed = set([ps.stem(word) for word in unique_lower])
  print(f"Unikalne {LANGUAGES[l]} (ze stemmingiem): {len(lower_stemmed)}")
```

```
Unikalne Angielski (bez steamingu): 2067
Unikalne Angielski (ze stemmingiem): 1644
Unikalne Polski (bez steamingu): 3133
Unikalne Polski (ze stemmingiem): 3100
Unikalne Włoski (bez steamingu): 2476
Unikalne Włoski (ze stemmingiem): 2457
```

- policz ilosć znaków
- policz ilosć unikalnych znaków

[109]:
```python
from collections import Counter

for l in LANGUAGES:
  c = Counter(TXT[l])
  chars = sum(c.values())
  print(f"Ilość dla {LANGUAGES[l]}: {chars}")
  print(f"Ilość unikalnych dla {LANGUAGES[l]}: {len(c.keys())}")
```

```
Ilość dla Angielski: 90208
Ilość unikalnych dla Angielski: 93
Ilość dla Polski: 75013
Ilość unikalnych dla Polski: 98
Ilość dla Włoski: 74078
Ilość unikalnych dla Włoski: 92
```

- policz ilosć zdań zdań
- policz ilosć unikalnych zdań

[110]:
```
for l in LANGUAGES:
    tokenized = nltk.tokenize.sent_tokenize(TXT[l])
    uniqe = set(tokenized)
    print(f'Liczba zdań dla {LANGUAGES[l]}: {len(tokenized)}')
    print(f'Liczba unikalnych zdań dla {LANGUAGES[l]}: {len(uniqe)}')
```

```
Liczba zdań dla Angielski: 1586
Liczba unikalnych zdań dla Angielski: 1523
Liczba zdań dla Polski: 1371
Liczba unikalnych zdań dla Polski: 1330
Liczba zdań dla Włoski: 1220
Liczba unikalnych zdań dla Włoski: 1184
```

- podaj min, max, średnią oraz medianę ilości znaków w słowie

[111]:
```
import numpy as np
for l in LANGUAGES:
    length = [len(w) for w in lowercase(TXT[l])]
    print(f"Minimalna ilość znaków dla {LANGUAGES[l]}:", min(length))
    print(f"Maksymalna ilość znaków dla {LANGUAGES[l]}", max(length))
    print(f"Średnia ilość znaków dla {LANGUAGES[l]}", np.mean(length))
    print(f"Mediana ilości znaków dla {LANGUAGES[l]}", np.median(length))
```

```
Minimalna ilość znaków dla Angielski: 1
Maksymalna ilość znaków dla Angielski 19
Średnia ilość znaków dla Angielski 4.016314270569527
Mediana ilości znaków dla Angielski 4.0
Minimalna ilość znaków dla Polski: 1
Maksymalna ilość znaków dla Polski 19
Średnia ilość znaków dla Polski 5.370293265749457
Mediana ilości znaków dla Polski 5.0
Minimalna ilość znaków dla Włoski: 1
Maksymalna ilość znaków dla Włoski 27
Średnia ilość znaków dla Włoski 4.529555293191657
Mediana ilości znaków dla Włoski 4.0
```

- podaj min, max, średnią oraz medianę ilości słów w zdaniu, znajdz najkrotsze i najdluzsze zdania

[112]:
```
for l in LANGUAGES:
    tokenized = nltk.tokenize.sent_tokenize(TXT[l])
    words_list = []
    words_count = []
    for sentence in tokenized:
        words_list.append(list(lowercase(sentence)))
        words_count.append(len(list(lowercase(sentence))))

    index_min = np.argmin(words_count)
    index_max = np.argmax(words_count)
```

```
    print(f"Minimalna ilość słów dla {LANGUAGES[l]}:", min(words_count))
    print(f"Maksymalna ilość słów dla {LANGUAGES[l]}", max(words_count))
    print(f"Średnia ilość słów dla {LANGUAGES[l]}", np.mean(words_count))
    print(f"Mediana ilości słów dla {LANGUAGES[l]}", np.median(words_count))
    print(f"Najkrótsze zdanie dla {LANGUAGES[l]}: ", " ".
↪join(words_list[index_min]), ".")
    print(f"Najdłuższe zdanie dla {LANGUAGES[l]}: ", " ".
↪join(words_list[index_max]), ".")
```

```
Minimalna ilość słów dla Angielski: 0
Maksymalna ilość słów dla Angielski 90
Średnia ilość słów dla Angielski 10.705548549810844
Mediana ilości słów dla Angielski 9.0
Najkrótsze zdanie dla Angielski:   .
Najdłuższe zdanie dla Angielski:  But in herself alone she is more important
than all the hundreds of you other roses because it is she that I have watered
because it is she that I have put under the glass globe because it is she that I
have sheltered behind the screen because it is for her that I have killed the
caterpillars except the two or three that we saved to become butterflies because
it is she that I have listened to when she grumbled or boasted or ever sometimes
when she said nothing .
Minimalna ilość słów dla Polski: 1
Maksymalna ilość słów dla Polski 44
Średnia ilość słów dla Polski 8.058351568198395
Mediana ilości słów dla Polski 6.0
Najkrótsze zdanie dla Polski:  Narysowałem .
Najdłuższe zdanie dla Polski:  Ale na twojej planecie mogłeś przesunąć krzesełko
o parę kroków i oglądać zachód słońca tyle razy ile chciałeś Pewnego dnia
oglądałem zachód słońca czterdzieści trzy razy powiedział Mały Książę a w chwilę
później dodał Wiesz gdy jest bardzo smutno to kocha się zachody słońca .
Minimalna ilość słów dla Włoski: 1
Maksymalna ilość słów dla Włoski 75
Średnia ilość słów dla Włoski 10.413934426229508
Mediana ilości słów dla Włoski 8.0
Najkrótsze zdanie dla Włoski:  Cosa .
Najdłuższe zdanie dla Włoski:  E sentendosi un po triste al pensiero dels uo
piccolo pianeta abbandonato si azzardo a sollecitare una grazia al re Vorrei
tatno vedere u tramonto Fatemi questo piacere Ordinate al sole di tramontare Se
ordinassi a un generale di volare da un fiore all altro come una farfalla o di
scrivere una tragedia o di trasformarsi in un uccello marino e se il generale
non eseguisse l ordine ricevuto chi avrebe torto lui o io .
```

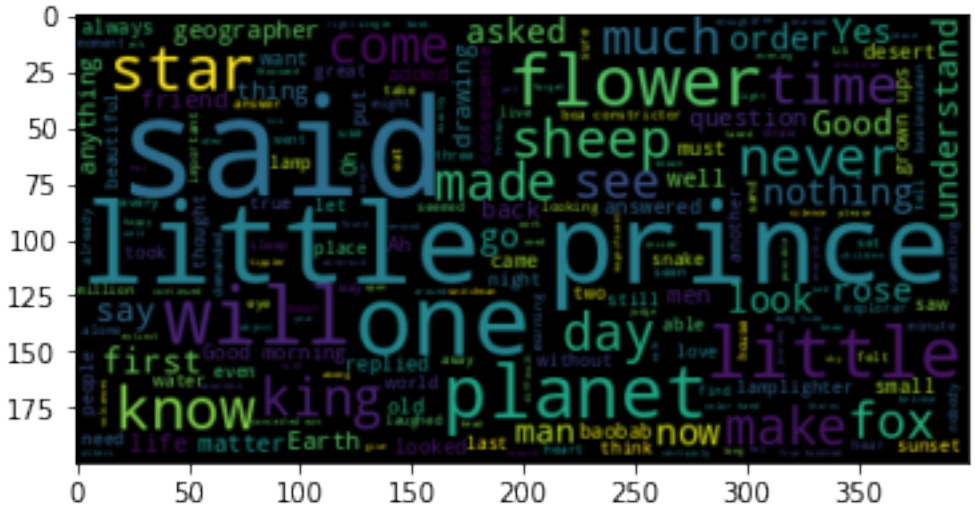
- wygeneruj word cloud (normalnie i po usunięciu stopwordów)

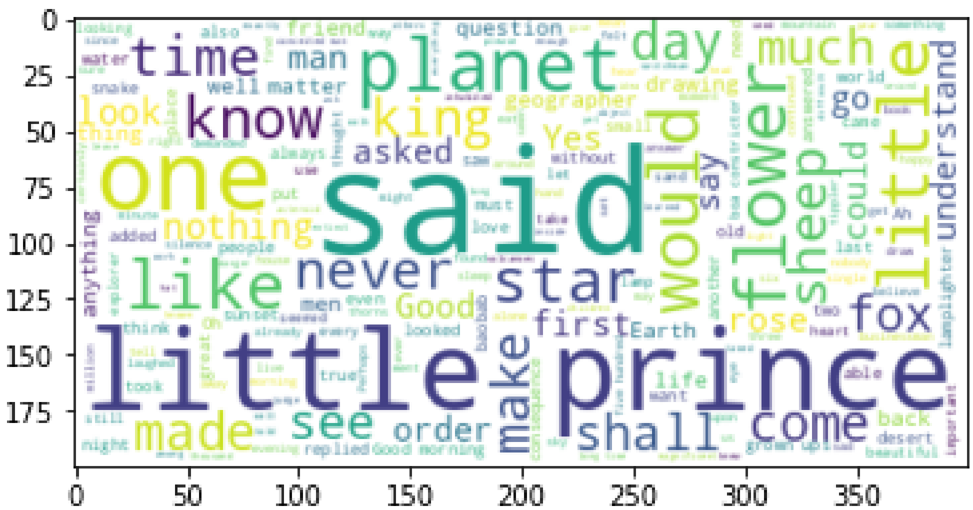
[113]:
```
!curl https://raw.githubusercontent.com/bieli/stopwords/master/polish.stopwords.
↪txt > outputs/polish_stopwords.txt
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2016  100  2016    0     0   8064      0 --:--:-- --:--:-- --:--:--  8064
```

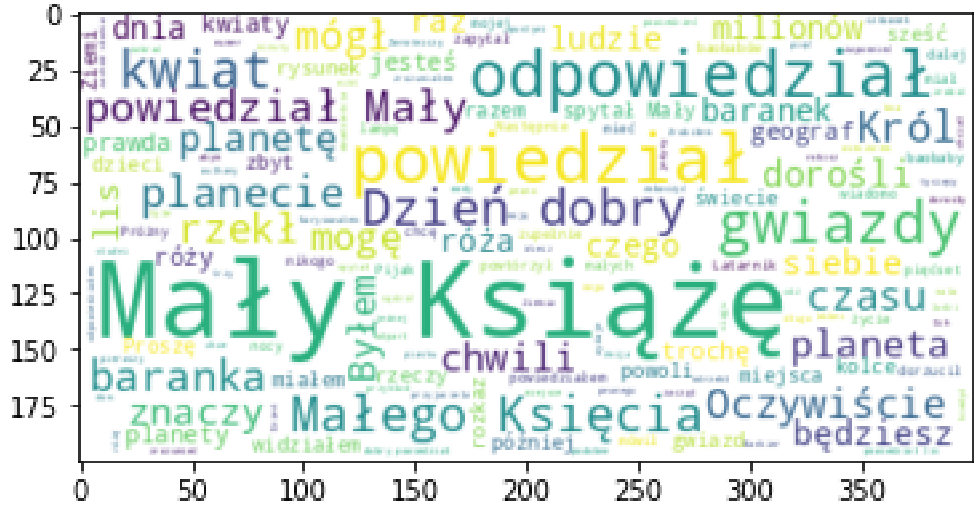
[114]:
```
with open(f"outputs//polish_stopwords.txt", "r") as f:
    polish_stopwords = [line.rstrip('\n') for line in f]
```

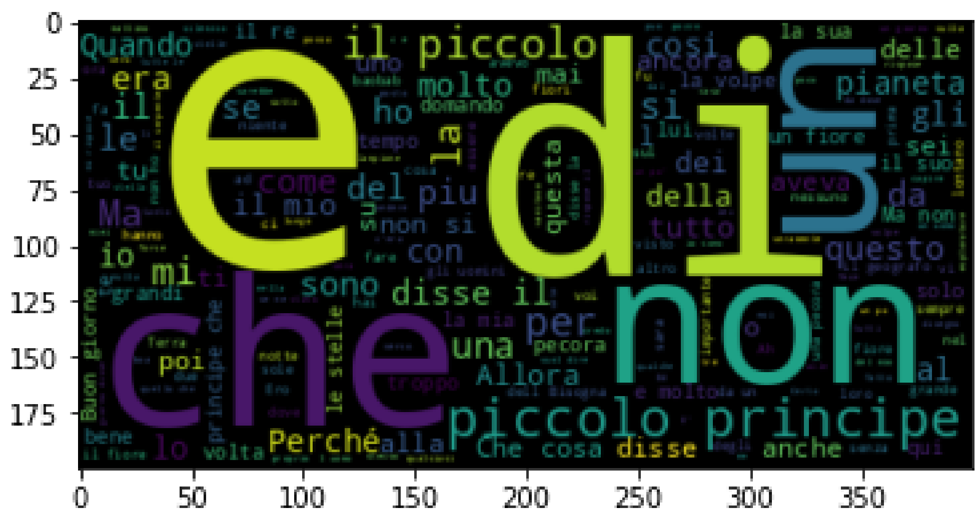
[115]:
```
import numpy as np
import pandas as pd
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt

for l in LANGUAGES:
    wordcloud = WordCloud().generate(TXT[l])
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.show()

    if (l=="pl"):
        stopwords = polish_stopwords
    elif(l=="en"):
        stopwords = nltk.corpus.stopwords.words('english')
    elif(l=="it"):
        stopwords = nltk.corpus.stopwords.words('italian')

    wordcloud = WordCloud(stopwords=stopwords, background_color="white").
 ↪generate(TXT[l])
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.show()
```

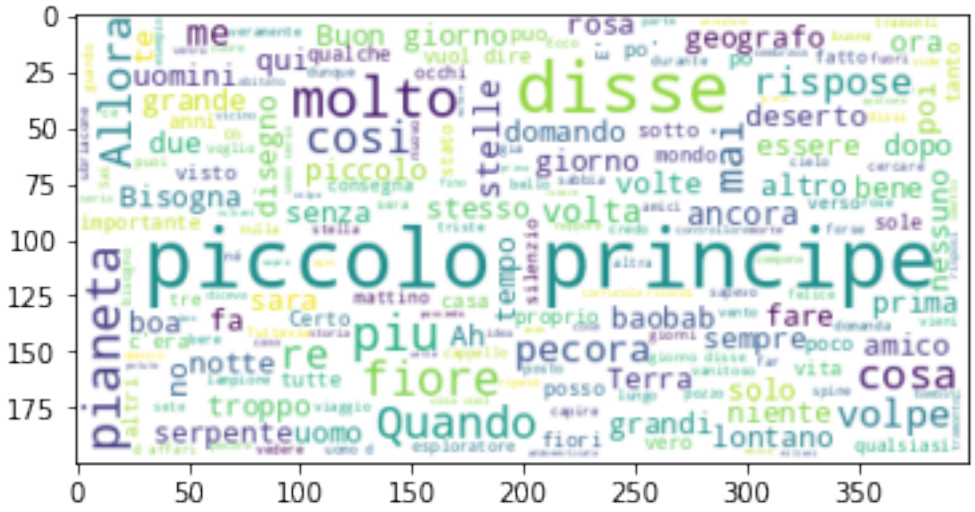



0
25
50
75
100
125
150
175
0
50
100
150
200
250
300
350
little prince
said
one
planet
time
know
king
like
never
star
would
flower
sheep
little
much
day
understand
could
made
see
order
shall
come
make
fox
rose
first
Good
nothing
look
asked
Yes
man
go
say
Earth
back


0
25
50
75
100
125
150
175
0
50
100
150
200
250
300
350
w
Mały Książę
na
nie
się
mnie
za
Lecz
który
jak
że
ponieważ
bardzo
jest
tak
odpowiedział
jeszcze
był
co
go
gdy
jego
są
aby
Ale
więc
już
jestem
dla
po
tym
sobie
będzie
tylko
gwiazdy
mam
powiedział
czy
mi
o
od
kwiat


Mały Książę
powiedział
odpowiedział
kwiat
powiedział Mały
Dzień dobry
gwiazdy
planecie
planetę
Król
rzekł
mogę
róża
czego
siebie
czasu
planeta
chwili
baranka
Małego Księcia
Oczywiście
znaczy
będziesz
dorośli
baranek
mógł
ludzie
milionów
dnia
kwiaty
raz
jesteś
prawda
lis
Byłem
trochę
róży
0
25
50
75
100
125
150
175
0
50
100
150
200
250
300
350


e
di
un
che
non
piccolo principe
il piccolo
disse il
Quando
così
delle
ancora
pianeta
molto
era
il
se
ho
si
gli
le
la
questa
dei
tu
come
del
piu
della
aveva
da
Ma
il mio
tutto
non si
con
questo
io
mi
sono
per
una
Allora
al
poi
Perché
alla
Che cosa
disse
anche
lo
volta
0
25
50
75
100
125
150
175
0
50
100
150
200
250
300
350



- wypisz 20 najbardziej popularnych słów (normalnie) (lowercase)

[116]:
```
from collections import OrderedDict
for l in LANGUAGES:
    print(f'Najbardziej popularne słowa dla {LANGUAGES[l]} ------')
    c = Counter(lowercase(TXT[l].lower()))
    items = c.most_common(20)
    print(OrderedDict(sorted(items, key=lambda t: -t[1])))
```

```
Najbardziej popularne słowa dla Angielski ------
OrderedDict([('the', 975), ('i', 545), ('to', 469), ('a', 413), ('and', 356),
('of', 336), ('that', 323), ('you', 312), ('is', 301), ('he', 297), ('it', 264),
('little', 258), ('said', 195), ('was', 193), ('prince', 184), ('in', 176),
('my', 164), ('not', 160), ('me', 153), ('but', 150)])
Najbardziej popularne słowa dla Polski ------
OrderedDict([('nie', 316), ('się', 306), ('to', 199), ('i', 196), ('na', 185),
('mały', 157), ('w', 156), ('książę', 142), ('jest', 125), ('z', 118), ('a',
84), ('że', 84), ('bardzo', 82), ('powiedział', 77), ('mi', 75), ('do', 75),
('o', 72), ('tak', 66), ('mnie', 60), ('lecz', 58)])
Najbardziej popularne słowa dla Włoski ------
OrderedDict([('e', 533), ('il', 427), ('che', 329), ('di', 313), ('un', 293),
('non', 282), ('la', 202), ('piccolo', 184), ('principe', 172), ('si', 152),
('a', 143), ('disse', 137), ('ma', 134), ('una', 134), ('per', 133), ('mi',
124), ('i', 107), ('le', 103), ('da', 97), ('l', 89)])
```

- wypisz 20 najbardziej popularnych słów (po usunięciu stopwordów) (lowercase)

[117]:
```
from collections import OrderedDict
for l in LANGUAGES:
    print(f'Najbardziej popularne słowa dla {LANGUAGES[l]} ------')
```

```
    if (l=="pl"):
        stopwords = polish_stopwords
    elif(l=="en"):
        stopwords = nltk.corpus.stopwords.words('english')
    elif(l=="it"):
        stopwords = nltk.corpus.stopwords.words('italian')
    text = lowercase(TXT[l].lower())
    without_stopwords = [word for word in text if not word in stopwords]
    c = Counter(without_stopwords)
    items = c.most_common(20)
    print(OrderedDict(sorted(items, key=lambda t: -t[1])))
```

```
Najbardziej popularne słowa dla Angielski ------
OrderedDict([('little', 258), ('said', 195), ('prince', 184), ('one', 140),
('planet', 69), ('would', 62), ('like', 58), ('flower', 54), ('good', 49),
('time', 46), ('never', 41), ('stars', 41), ('sheep', 41), ('made', 37),
('know', 37), ('come', 37), ('shall', 35), ('fox', 35), ('man', 34), ('much',
33)])
Najbardziej popularne słowa dla Polski ------
OrderedDict([('mały', 157), ('książę', 142), ('powiedział', 77),
('odpowiedział', 39), ('gwiazdy', 30), ('dzień', 27), ('dobry', 27), ('małego',
25), ('księcia', 23), ('kwiat', 23), ('lis', 22), ('będziesz', 20), ('baranka',
19), ('chwili', 19), ('planecie', 19), ('słońca', 19), ('król', 19), ('czasu',
18), ('spytał', 18), ('oczywiście', 17)])
Najbardziej popularne słowa dla Włoski ------
OrderedDict([('piccolo', 184), ('principe', 172), ('disse', 137), ('molto', 72),
('piu', 61), ('pianeta', 56), ('fiore', 50), ('cosa', 46), ('giorno', 46),
('cosi', 44), ('quando', 43), ('mai', 40), ('allora', 39), ('re', 36),
('rispose', 35), ('pecora', 33), ('uomo', 32), ('volpe', 32), ('me', 30),
('stelle', 28)])
```

- wypisz 20 najbardziej popularnych bigramów (normalnie)

[118]:
```
for l in LANGUAGES:
    bigrams = Counter(list(nltk.bigrams(lowercase(TXT[l]))))
    print(f'Najbardziej popularne biogramy dla {LANGUAGES[l]}\n {bigrams.
 ↪most_common(20)}')
```

```
Najbardziej popularne biogramy dla Angielski
 [(('little', 'prince'), 183), (('the', 'little'), 152), (('said', 'the'), 93),
(('of', 'the'), 69), (('It', 'is'), 60), (('in', 'the'), 58), (('I', 'have'),
58), (('I', 'am'), 53), (('said', 'to'), 43), (('I', 'was'), 39), (('it', 'is'),
38), (('to', 'me'), 37), (('a', 'little'), 36), (('he', 'said'), 36), (('to',
'the'), 35), (('that', 'I'), 34), (('did', 'not'), 32), (('of', 'a'), 31),
(('I', 'shall'), 30), (('the', 'stars'), 30)]
Najbardziej popularne biogramy dla Polski
 [(('Mały', 'Książę'), 142), (('mi', 'się'), 27), (('Małego', 'Księcia'), 22),
(('Dzień', 'dobry'), 21), (('powiedział', 'Mały'), 20), (('się', 'na'), 16),
```

```
(('nie', 'ma'), 15), (('nie', 'jest'), 14), (('się', 'w'), 14), (('nic', 'nie'),
13), (('to', 'nie'), 12), (('spytał', 'Mały'), 12), (('nigdy', 'nie'), 11),
(('się', 'że'), 11), (('Książę', 'nie'), 8), (('nie', 'wiadomo'), 8),
(('odpowiedział', 'Mały'), 8), (('dobry', 'powiedział'), 8), (('powiedział',
'sobie'), 8), (('na', 'świecie'), 7)]
Najbardziej popularne biogramy dla Włoski
 [(('piccolo', 'principe'), 169), (('il', 'piccolo'), 117), (('disse', 'il'),
79), (('Il', 'piccolo'), 30), (('il', 'mio'), 25), (('che', 'non'), 24), (('di',
'un'), 24), (('un', 'po'), 22), (('non', 'e'), 21), (('la', 'volpe'), 21),
(('il', 're'), 20), (('e', 'un'), 19), (('Buon', 'giorno'), 19), (('un',
'fiore'), 18), (('la', 'sua'), 17), (('il', 'suo'), 17), (('le', 'stelle'), 17),
(('principe', 'che'), 16), (('disse', 'la'), 16), (('la', 'mia'), 15)]
```

- wypisz 20 najbardziej popularnych bigramów (normalnie)

[119]:
```
for l in LANGUAGES:
    if (l=="pl"):
        stopwords = polish_stopwords
    elif(l=="en"):
        stopwords = nltk.corpus.stopwords.words('english')
    elif(l=="it"):
        stopwords = nltk.corpus.stopwords.words('italian')
    text = lowercase(TXT[l])
    without_stopwords = [word for word in text if not word in stopwords]
    bigrams = Counter(list(nltk.bigrams(without_stopwords)))
    print(f'Najbardziej popularne biogramy dla {LANGUAGES[l]}\n {bigrams.
 ↪most_common(20)}')
```

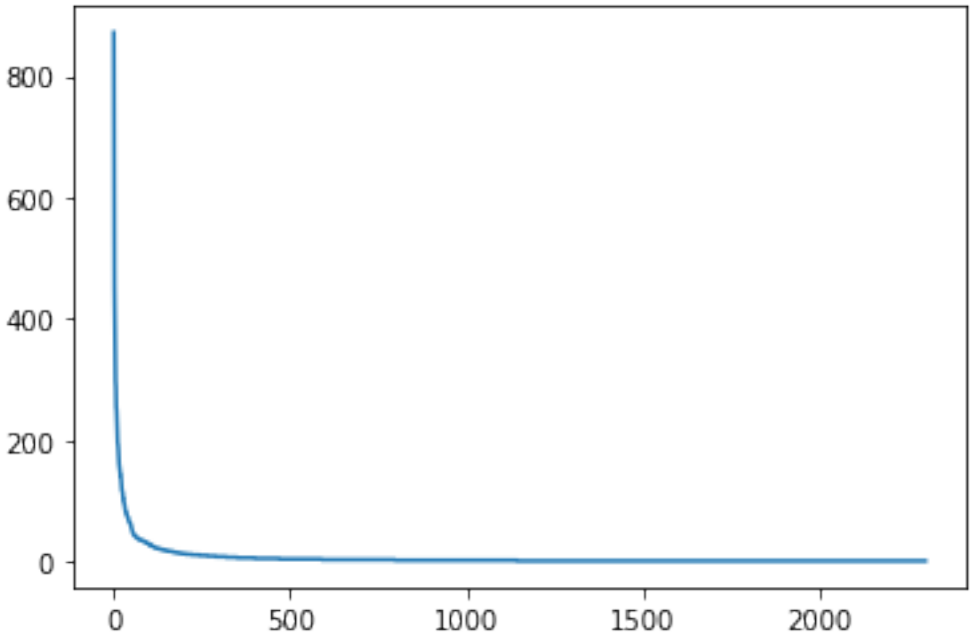
```
Najbardziej popularne biogramy dla Angielski
 [(('little', 'prince'), 183), (('said', 'little'), 44), (('And', 'I'), 32),
(('I', 'shall'), 31), (('The', 'little'), 27), (('I', 'said'), 26), (('Good',
'morning'), 21), (('said', 'I'), 19), (('grown', 'ups'), 18), (('I', 'know'),
17), (('But', 'I'), 15), (('said', 'fox'), 15), (('I', 'made'), 13), (('So',
'I'), 13), (('I', 'could'), 13), (('And', 'little'), 11), (('prince', 'I'), 11),
(('asked', 'little'), 11), (('morning', 'said'), 11), (('matters',
'consequence'), 9)]
Najbardziej popularne biogramy dla Polski
 [(('Mały', 'Książę'), 142), (('powiedział', 'Mały'), 23), (('Małego',
'Księcia'), 22), (('Dzień', 'dobry'), 21), (('spytał', 'Mały'), 12), (('Co',
'znaczy'), 10), (('odpowiedział', 'Mały'), 8), (('dobry', 'powiedział'), 8),
(('Po', 'chwili'), 7), (('rzekł', 'Mały'), 7), (('powiedział', 'lis'), 7),
(('zachód', 'słońca'), 6), (('człowiekiem', 'poważnym'), 6), (('Nie', 'mogę'),
6), (('tysiąc', 'mil'), 5), (('To', 'prawda'), 5), (('pięćset', 'milionów'), 5),
(('powtórzył', 'Mały'), 5), (('Kim', 'jesteście'), 5), (('Mały', 'przyjacielu'),
5)]
Najbardziej popularne biogramy dla Włoski
 [(('piccolo', 'principe'), 169), (('disse', 'piccolo'), 43), (('Il',
'piccolo'), 30), (('Buon', 'giorno'), 19), (('disse', 'volpe'), 15), (('Che',
```

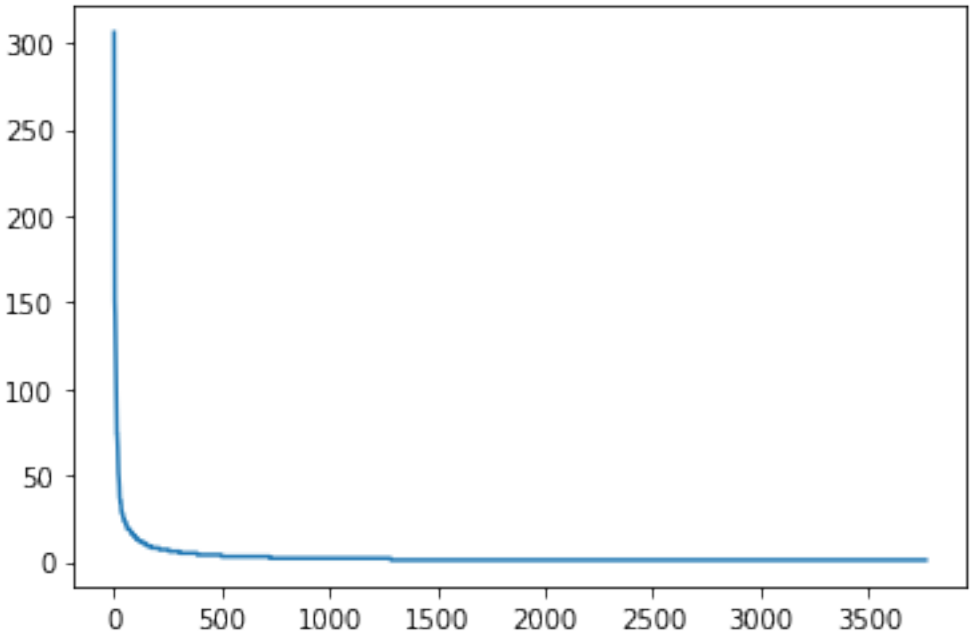
```
'cosa'), 14), (('giorno', 'disse'), 11), (('E', 'molto'), 10), (('vuol',
'dire'), 9), (('E', 'piccolo'), 8), (('domando', 'piccolo'), 8), (('uomo', 'd'),
8), (('d', 'affari'), 8), (('disse', 'geografo'), 8), (('Non', 'mai'), 7),
(('Che', 'cos'), 7), (('uomo', 'serio'), 7), (('campana', 'vetro'), 7),
(('disse', 're'), 7), (('cosa', 'vuol'), 7)]
```

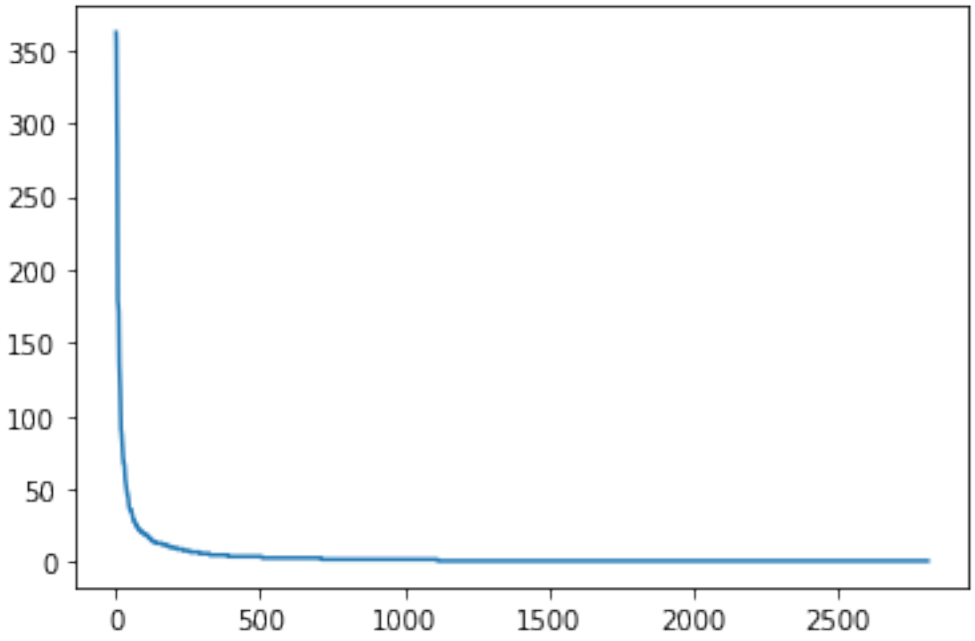
- narysuj wykres częstotliwości słów (histogram lub linie) w taki sposób żeby był czytelny, wypróbuj skali logarytmicznej dla osi x (ale na razie nie dla y), usuwanie słów poniżej limitu wystąpień itp.

[120]:
```
import matplotlib.pyplot as plt
from collections import OrderedDict

def freq_list(g, top=None):
  c = Counter(g)

  if top is None:
     items = c.items()
  else:
     items = c.most_common(top)

  return OrderedDict(sorted(items, key=lambda t: -t[1]))

def rang_freq_with_labels(name, g, top=None):
   freq = freq_list(g, top)

   plt.figure(figsize=(12, 3))
   plt.ylabel('liczba wystąpień')

   plt.bar(freq.keys(), freq.values())

   plt.show()


def rang_freq(name, g):
   freq = freq_list(g)

   plt.figure().clear()
   plt.plot(range(1, len(freq.values())+1), freq.values())

   plt.show()
```

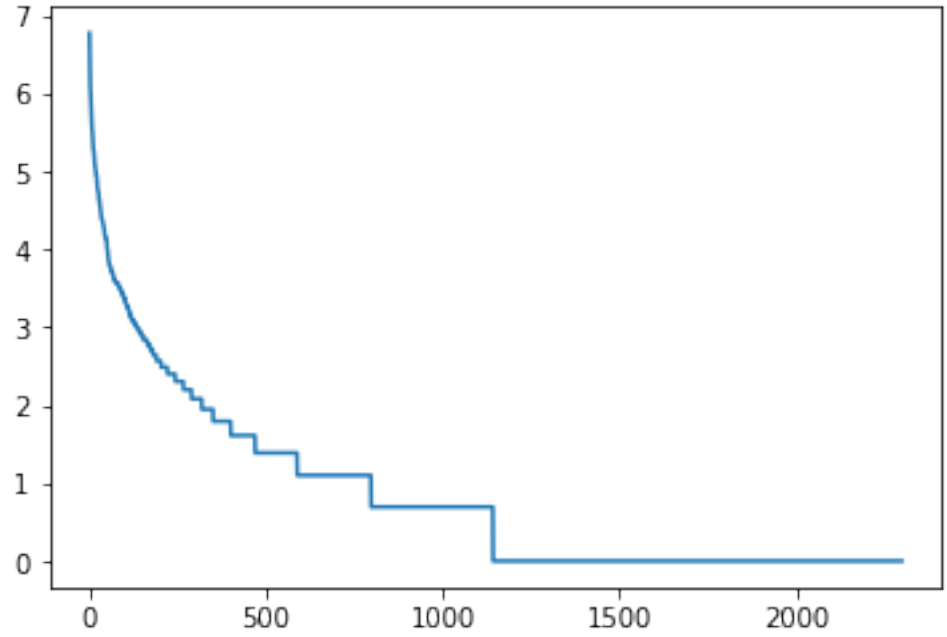
[121]:
```
for l in LANGUAGES:
    rang_freq('pt-words', lowercase(TXT[l]))
```

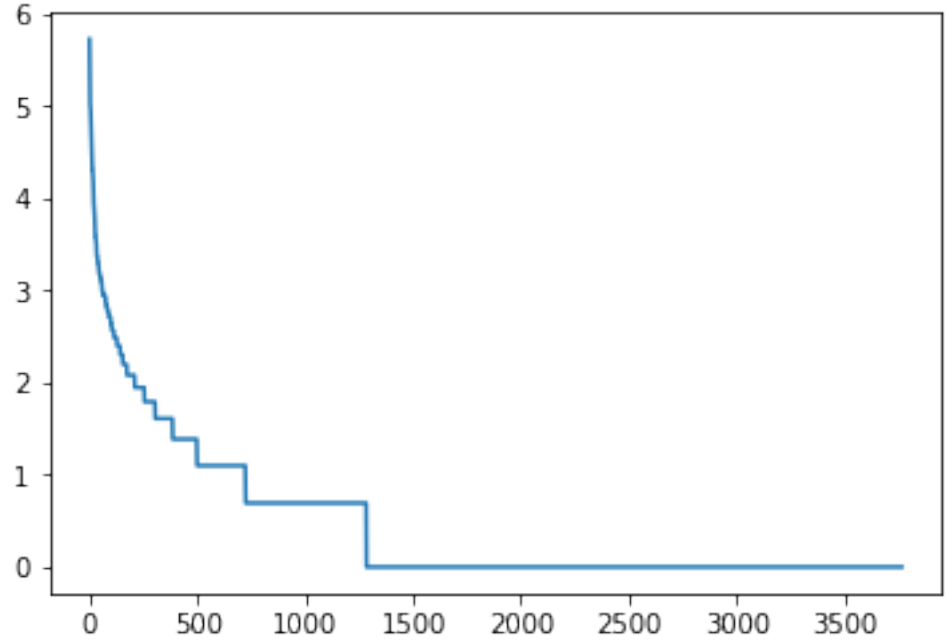

800
600
400
200
0
0
500
1000
1500
2000

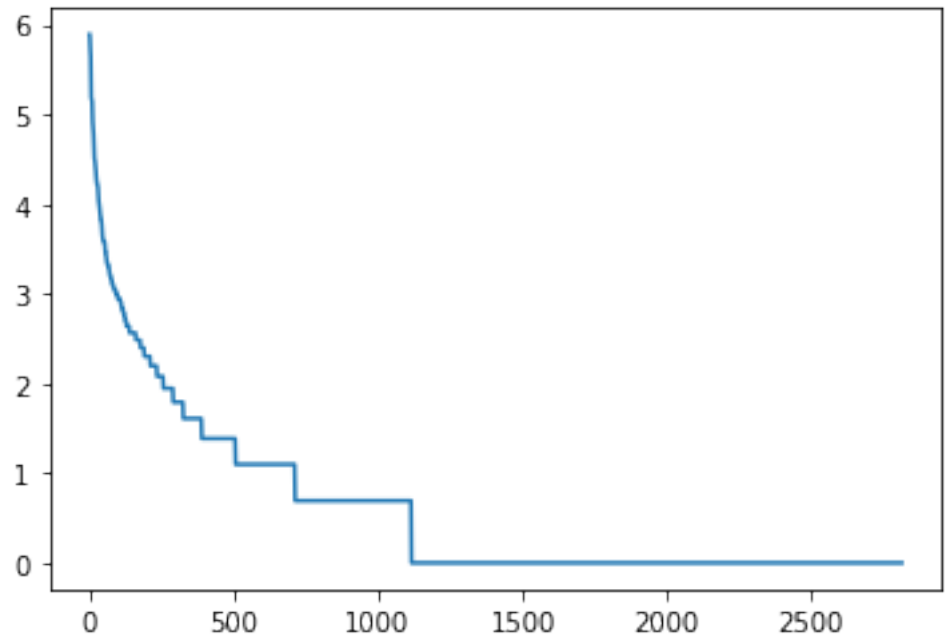

300
250
200
150
100
50
0
0
500
1000
1500
2000
2500
3000
3500



[122]:
```
from math import log
def rang_logaritm_freq(name, g):
   freq = freq_list(g)
   plt.figure().clear()
   plt.plot(range(1, len(freq.values())+1), [log(y) for y in freq.values()])

   plt.show()
for l in LANGUAGES:
    rang_logaritm_freq('pt-words', lowercase(TXT[l]))
```

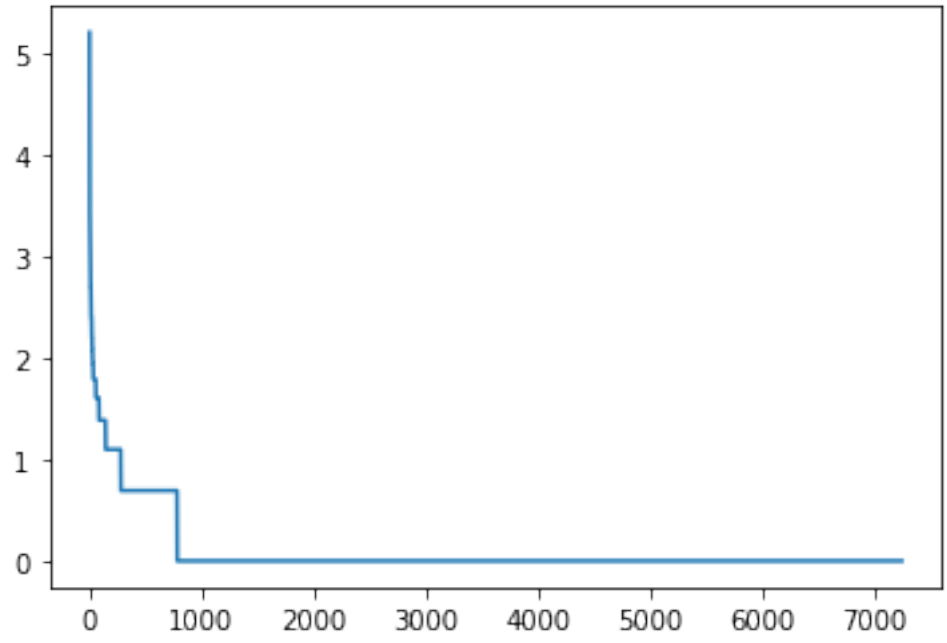

7
6
5
4
3
2
1
0
0
500
1000
1500
2000

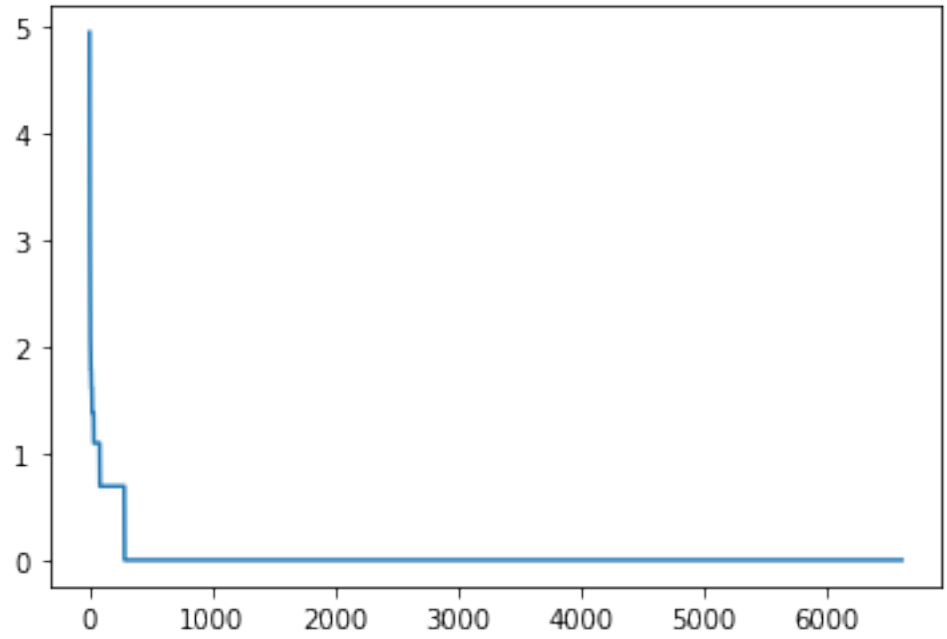

6
5
4
3
2
1
0
0
500
1000
1500
2000
2500
3000
3500

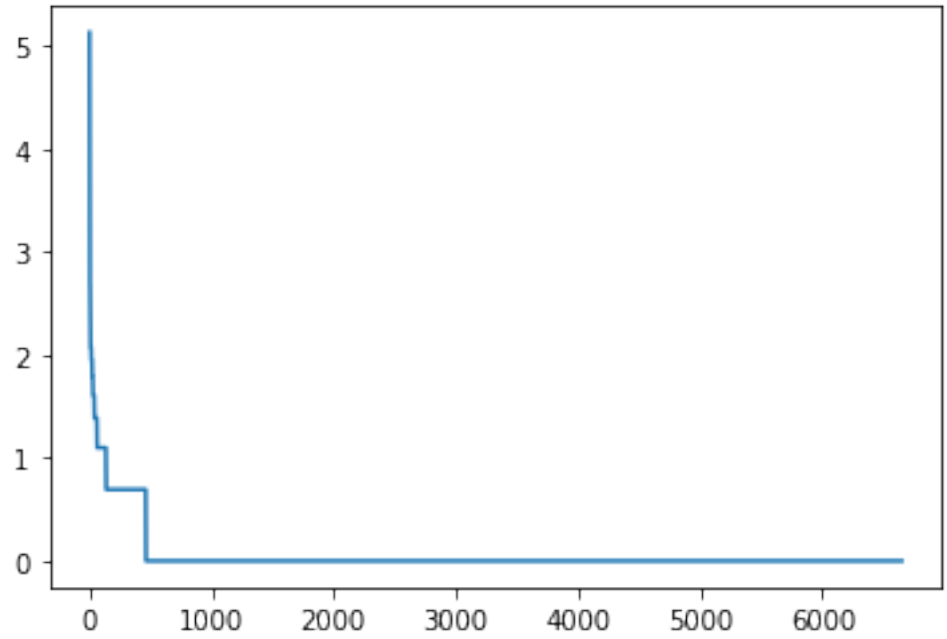


- punkt jak wyżej, tylko dla bigramów

[125]:
```
for l in LANGUAGES:
    if (l=="pl"):
        stopwords = polish_stopwords
    elif(l=="en"):
        stopwords = nltk.corpus.stopwords.words('english')
    elif(l=="it"):
        stopwords = nltk.corpus.stopwords.words('italian')
    text = lowercase(TXT[l])
    without_stopwords = [word for word in text if not word in stopwords]
    bigrams = Counter(list(nltk.bigrams(without_stopwords)))
    rang_logaritm_freq('pt-bigrams', bigrams)
```


5
4
3
2
1
0
0
1000
2000
3000
4000
5000
6000
7000


5
4
3
2
1
0
0
1000
2000
3000
4000
5000
6000



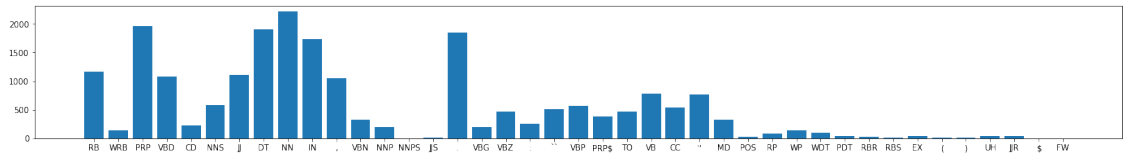
- narysuj wykres barplot dla części mowy (PART OF SPEECH TAGS, tylko pierwszy stopień zagłębienia)

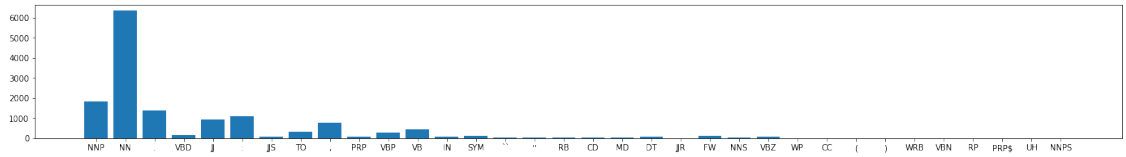
[128]:
```
import nltk
nltk.download('averaged_perceptron_tagger')
```

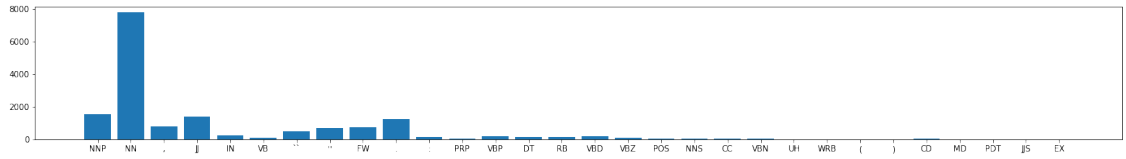
```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/maciej/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

[128]: True

[129]:
```
for l in LANGUAGES:
    text = nltk.tokenize.word_tokenize(TXT[l])
    tags = nltk.pos_tag(text)
    c = Counter([tag[1] for tag in tags])
    plt.figure(figsize=(24, 3))
    plt.bar(c.keys(), c.values())
```







- dla próbki 10000 zdań sprawdź jak często langdetect https://pypi.org/project/langdetect/ się myli i w jaki sposób.

[143]:
```
from langdetect import detect
import random
import nltk

def detect_lng(sentence, lng):
    try:
        detector = detect(sentence)
        if detector == lng:
            return True
        else:
            print(f'[{lng}] {sentence}')
            return False
    except Exception:
        pass
counters = {}
for l in LANGUAGES:
    counters[l] = 0
    for sentence in random.sample(nltk.tokenize.sent_tokenize(TXT[l]), 1000):
        if(detect_lng(sentence, l)):
            counters[l] +=1
for l in LANGUAGES:
    print(f"Dla {LANGUAGES[l]} poprawnie wykryto {counters[l]}/1000")
```

```
[en] Niepoprawne To nobody."
[en] Niepoprawne "Anything you like .
[en] Niepoprawne Hum!
[en] Niepoprawne I shall show you how happy I am!
[en] Niepoprawne "Eh?
[en] Niepoprawne "I forbid you to do so."
[en] Niepoprawne "Ah!
```

[en] Niepoprawne . .
Look at my planet.
[en] Niepoprawne I blinked my
eyes hard.
[en] Niepoprawne "Hum!
[en] Niepoprawne Hum!"
[en] Niepoprawne "Ah!"
[en] Niepoprawne "Oh!
[en] Niepoprawne Yawn again!
[en] Niepoprawne He took one
step.
[en] Niepoprawne . .

"Here it is.
[en] Niepoprawne Good evening."
[en] Niepoprawne . ."
I said nothing.
[en] Niepoprawne You will suffer.
[en] Niepoprawne "I am so unhappy."
[en] Niepoprawne You cannot even travel .
[en] Niepoprawne . ."
"No.
[en] Niepoprawne "No."
[en] Niepoprawne "Do you hear?"
[en] Niepoprawne "Bees?"
[en] Niepoprawne "Yes."
[en] Niepoprawne "I order you to
yawn.
[en] Niepoprawne "One could not die for you.
[en] Niepoprawne . ." And I felt him to be more fragile still.
[en] Niepoprawne I am glad!"
[en] Niepoprawne To forget a friend is
sad.
[en] Niepoprawne he said.
[en] Niepoprawne "I am looking
for friends.
[en] Niepoprawne "But you must not forget it.
[en] Niepoprawne I demanded.
[en] Niepoprawne You are responsible for your rose .
[en] Niepoprawne "No."
[en] Niepoprawne I said nothing.
[en] Niepoprawne "Oh, no.
[en] Niepoprawne "Hum!
[en] Niepoprawne What a queer idea!"
[en] Niepoprawne But he did not answer me.
[en] Niepoprawne "Yes."
[en] Niepoprawne I felt awkward and
blundering.

[en] Niepoprawne "Please excuse me .
[en] Niepoprawne Good evening."
[en] Niepoprawne I have so much to do!
[en] Niepoprawne "Do not go.
[en] Niepoprawne "Wait?
[en] Niepoprawne "Ah!"
[en] Niepoprawne FIfteen and seven make twenty-two.
[en] Niepoprawne I have no time for loafing.
[en] Niepoprawne "Ah!
[en] Niepoprawne "No, no, no!
[en] Niepoprawne "Oh, no.
[en] Niepoprawne "Ah!
[en] Niepoprawne Hum!"
[en] Niepoprawne "I
hunt chickens; men hunt me.
[en] Niepoprawne . . Do not come."
[en] Niepoprawne Now go!"
[en] Niepoprawne I am very old.
[en] Niepoprawne "Hum!
[en] Niepoprawne he demanded, thunderstruck.
[en] Niepoprawne "Ah!
[en] Niepoprawne "Be my friends.
[en] Niepoprawne "Yes?"
[en] Niepoprawne You are an explorer!
[pl] Niepoprawne I oboje umilkli.
[pl] Niepoprawne O, tak!
[pl] Niepoprawne - Narysuj mi baranka.
[pl] Niepoprawne - Ach tak?
[pl] Niepoprawne Ach!
[pl] Niepoprawne - Hm, hm!
[pl] Niepoprawne - Muszek?
[pl] Niepoprawne To mój samolot.
[pl] Niepoprawne - Nic.
[pl] Niepoprawne To nie jest baranek, to baran.
[pl] Niepoprawne On ma rogi.
[pl] Niepoprawne Narysuj mi baranka.
[pl] Niepoprawne To samolot.
[pl] Niepoprawne - Hm, hm!
[pl] Niepoprawne Ziarna baobabu.
[pl] Niepoprawne To bardzo daleko.
[pl] Niepoprawne - Co?
[pl] Niepoprawne Tak.
[pl] Niepoprawne - Tak.
[pl] Niepoprawne O, tak!
[pl] Niepoprawne I oboje umilkli.
[pl] Niepoprawne - Hm, hm.
[pl] Niepoprawne - Po co?
[pl] Niepoprawne - Ach!

[pl] Niepoprawne - Tak jest.
[pl] Niepoprawne Narysuj mi baranka.
[pl] Niepoprawne Ilu ma braci?
[pl] Niepoprawne Mam kolce.
[pl] Niepoprawne - A oto mój sekret.
[pl] Niepoprawne - Co?
[pl] Niepoprawne U pana jest bardzo zimno.
[pl] Niepoprawne Nie zostawiajcie mnie wtedy w moim smutku:
bądźcie tak mili i napiszcie mi szybko, że wrócił…

Průměr:

ZhodnoťSlabéUjde toDobrýSkvělýÚžasný
Tvé hlasování: Žádná Průměr: 4.5 (785 hlasů)

Mazais princis
o úroveň výš
Mažasis princas

Verze pro tisk
Přidat komentář
1237010x přečteno
Zaslat e-mailem
[pl] Niepoprawne Och!
[pl] Niepoprawne I taka naiwna.
[pl] Niepoprawne Rysunek numer 1.
[pl] Niepoprawne Posiadam je.
[pl] Niepoprawne - Tak.
[it] Niepoprawne "Ah!
[it] Niepoprawne "Ah!
[it] Niepoprawne Era caduta la note.
[it] Niepoprawne "Si.
[it] Niepoprawne "Perché?"
[it] Niepoprawne Che buffa idea!"

[it] Niepoprawne Ero commosso.
[it] Niepoprawne Di stelel".
[it] Niepoprawne "Ah!"
[it] Niepoprawne Hum!"
[it] Niepoprawne Era dolce come una festa.
[it] Niepoprawne Non si sa mai.
[it] Niepoprawne Ma no!
[it] Niepoprawne "Perché?"
[it] Niepoprawne Allora grido: "Come?
[it] Niepoprawne Avevo bevuto.
[it] Niepoprawne "Oh!
[it] Niepoprawne Quanto pesa?
[it] Niepoprawne "Ebbene?"
[it] Niepoprawne vergogna.
[it] Niepoprawne "Da dove vieni?"
[it] Niepoprawne Si sedette.
[it] Niepoprawne E poi, guarda!
[it] Niepoprawne "Non ho pu niente da fare qui", disse al re.
[it] Niepoprawne "Ah!
[it] Niepoprawne Le possiedo".
[it] Niepoprawne Non c'e da prendersela.
[it] Niepoprawne Ma non si sa ma".
[it] Niepoprawne "No".
[it] Niepoprawne Perché e la mia rosa".
[it] Niepoprawne Ma questo bisogna perdonarmelo.
[it] Niepoprawne "E' la.
[it] Niepoprawne "Di mosche?"
[it] Niepoprawne "Ma no.
[it] Niepoprawne Ed era vero.
[it] Niepoprawne "Si".
[it] Niepoprawne Non poté proseguire.
[it] Niepoprawne "Perché?
[it] Niepoprawne "Ma piangerai!"
[it] Niepoprawne Non lasciatemi cosi triste: scrivetemi subito che e ritornato…

Průměr:

ZhodnoťSlabéUjde toDobrýSkvělýÚžasný
Tvé hlasování: Žádná Průměr: 4.2 (86 hlasů)

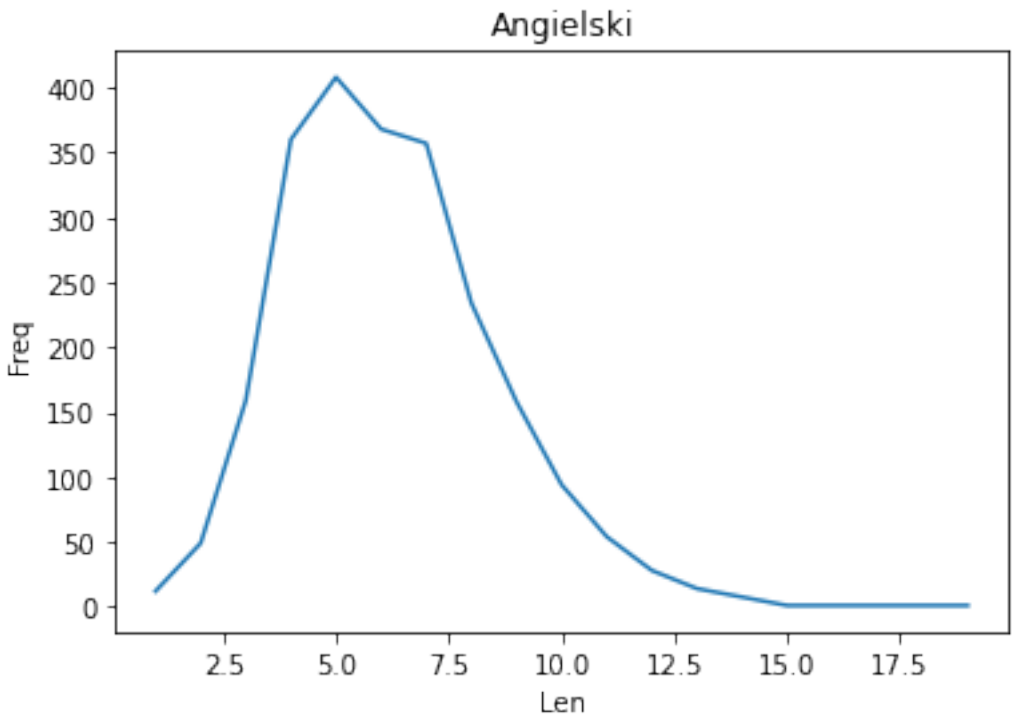
Hoàng Tử Bé
o úroveň výš
Kis herceg

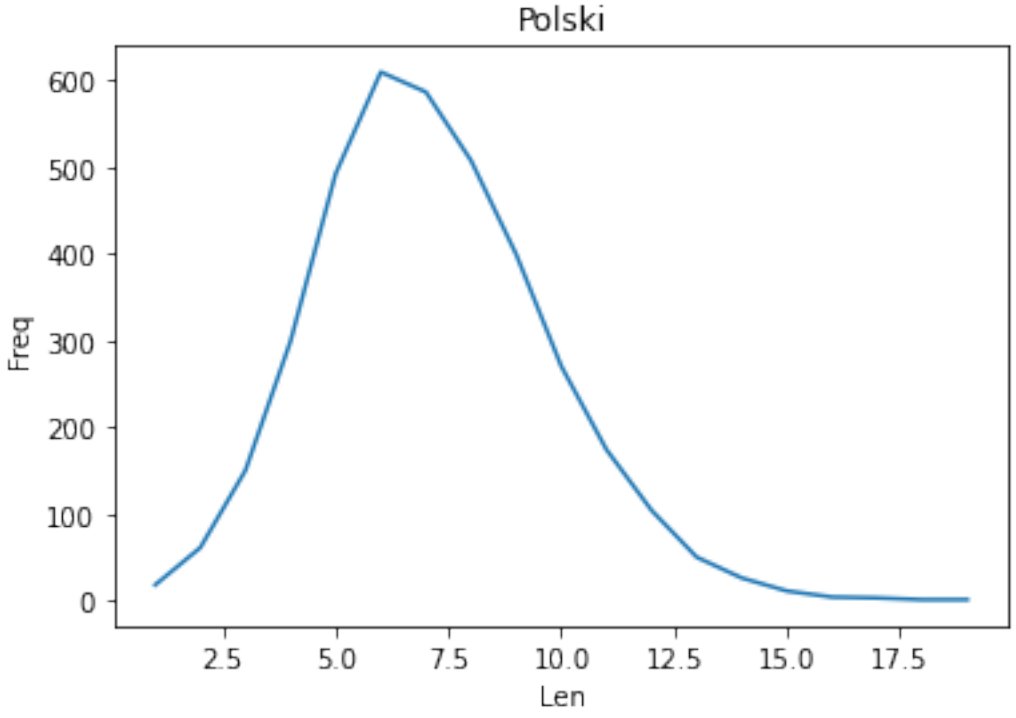
Verze pro tisk
Přidat komentář
208995x přečteno
Zaslat e-mailem
[it] Niepoprawne E un ordine".
[it] Niepoprawne hem!
[it] Niepoprawne "Niente.
[it] Niepoprawne E un aeroplano.
[it] Niepoprawne Respiravo bene.
[it] Niepoprawne "Si".
[it] Niepoprawne E un ariete.
[it] Niepoprawne Cadde dolcemente come cade un albero.
[it] Niepoprawne "Che cosa cercano?"
[it] Niepoprawne Disegnai dunque una museruola.
[it] Niepoprawne "Ah si?"
[it] Niepoprawne "Oh!
[it] Niepoprawne Ah!
[it] Niepoprawne "Di api?".
[it] Niepoprawne Ma vi domandano: "Che eta ha?
[it] Niepoprawne "E quando saranno?"
[it] Niepoprawne "E uno scambio".
[it] Niepoprawne Avrai dispiacere.
[it] Niepoprawne "Che cos'e questo grosso libro?"
[it] Niepoprawne E non e importante questo!"
[it] Niepoprawne "Hem!
[it] Niepoprawne "Hem!
[it] Niepoprawne "Perché bevi?"
[it] Niepoprawne Di stelle?"
[it] Niepoprawne Non credo niente!
[it] Niepoprawne "Be' !
[it] Niepoprawne "Legarla?
[it] Niepoprawne "Ah!"
[it] Niepoprawne "Che cos'e questa storia!
[it] Niepoprawne "Perché vendi questa roba? "
[it] Niepoprawne Posso…"
"Oh!
[it] Niepoprawne Tu sei un esploratore!
[it] Niepoprawne Hem!"

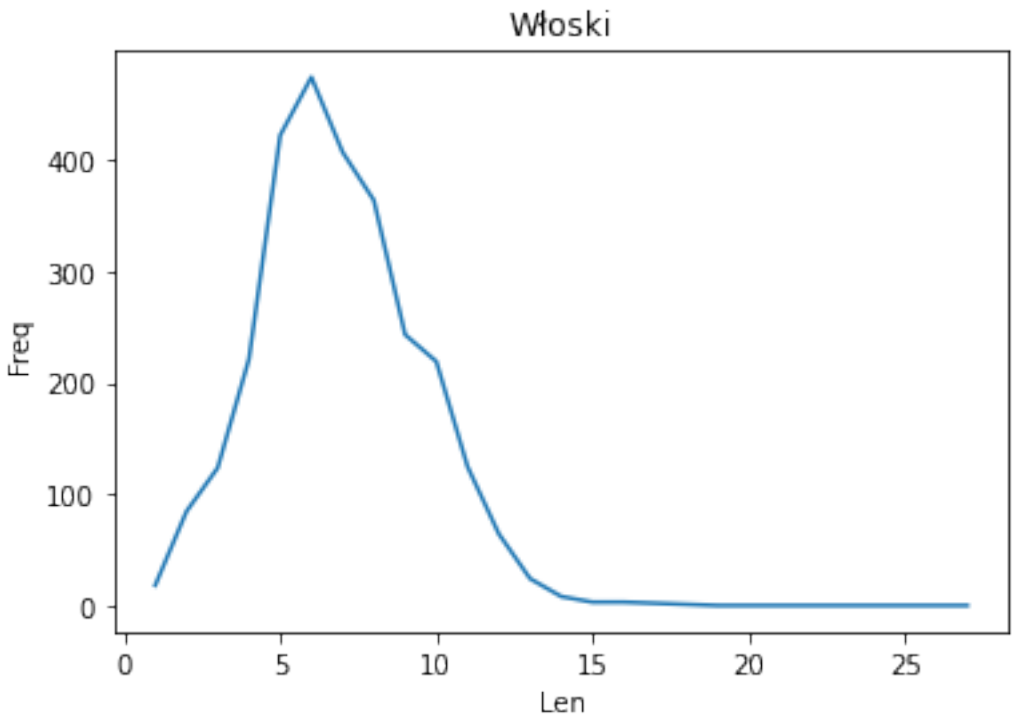
```
[it] Niepoprawne "Sara bello, sai.
[it] Niepoprawne "Ah!"
[it] Niepoprawne "Un fungo!"
[it] Niepoprawne Era grandioso.
[it] Niepoprawne "Addio", ripeté.
[it] Niepoprawne "Si.
[it] Niepoprawne "Che cosa?"
[it] Niepoprawne "Me ne vado".
[it] Niepoprawne hem!"
[it] Niepoprawne "Ah!
Dla Angielski poprawnie wykryto 923/1000
Dla Polski poprawnie wykryto 963/1000
Dla Włoski poprawnie wykryto 917/1000
```

- zilustruj prawo zipfa (px.line z zaznaczonymi punktami), bez skali logarytmicznej dla najbardziej popularnych 20 słów

[148]:
```python
def freq_vs_length(g, lng, top=None ):
    freq = freq_list(g)
    c = Counter([len(x) for x in freq.keys()])
    c = OrderedDict(sorted(c.items()))
    plt.figure()
    plt.plot([x for x in c.keys()], [y for y in c.values()])
    plt.title(f"{LANGUAGES[l]}")
    plt.xlabel("Len")
    plt.ylabel("Freq")
for l in LANGUAGES:
    freq_vs_length(lowercase(TXT[l]), lng=l)
```


Angielski
Freq
Len
0
50
100
150
200
250
300
350
400
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Polski
Freq
Len
0
100
200
300
400
500
600
2.5
5.0
7.5
10.0
12.5
15.0
17.5



KONIEC ZADANIA

0.2 ZADANIE 2

(20 punktów)

Znajdź teksty w języku polskim (mają składać sie po 5 osobnych dokumentów każdy, długości powinny być różne): - tekst prawny - tekst naukowy - tekst z polskiego z powieści (np. wolne lektury) - tekst z polskiego internetu (reddit, wykop, komentarze) - transkrypcja tekstu mówionego

ZADANIA: - zilustruj gunning fog index (oś y) i średnią długość zdania (oś x) na jednym wykresie dla wszystkich tekstów, domeny oznacz kolorami (px.scatter), dla języka polskiego traktuj wyrazy długie jako te powyżej 3 sylab, możesz użyć https://pyphen.org/ do liczenia sylab - zilustruj prawo Heaps'a dla wszystkich tekstów na jednym wykresie, domeny oznacz kolorami (px.scatter) - napisz wnioski (10-50 zdań)

START ZADANIA

KONIEC ZADANIA

0.3 WYKONANIE ZADAŃ

Zgodnie z instrukcją 01_Kodowanie_tekstu.ipynb