P04

October 1, 2023

0.1 Práctica 4

[1]:
```
# Bibliotecas
from collections import Counter
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = [15, 6]
from re import sub
import numpy as np
import string

!pip install elotl
!pip install wordcloud
!pip install nltk
!pip install unidecode
from wordcloud import WordCloud
from elotl import corpus as elotl_corpus
import nltk
nltk.download('brown')
nltk.download('cess_esp')
nltk.download('stopwords')
from nltk.corpus import brown as brown
from nltk.corpus import cess_esp as cess
from nltk.corpus import stopwords
from unidecode import unidecode
axolotl = elotl_corpus.load("axolotl")
tsunkua = elotl_corpus.load("tsunkua")
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: elotl in /home/xbmu/.local/lib/python3.8/site-
packages (0.0.1.16)
Requirement already satisfied: importlib-resources in
/home/xbmu/.local/lib/python3.8/site-packages (from elotl) (6.1.0)
Requirement already satisfied: future in /home/xbmu/.local/lib/python3.8/site-
packages (from elotl) (0.18.3)
Requirement already satisfied: zipp>=3.1.0 in
/home/xbmu/.local/lib/python3.8/site-packages (from importlib-resources->elotl)
(3.17.0)
Defaulting to user installation because normal site-packages is not writeable
```

```
Requirement already satisfied: wordcloud in
/home/xbmu/.local/lib/python3.8/site-packages (1.9.2)
Requirement already satisfied: numpy>=1.6.1 in
/home/xbmu/.local/lib/python3.8/site-packages (from wordcloud) (1.24.4)
Requirement already satisfied: pillow in /home/xbmu/.local/lib/python3.8/site-
packages (from wordcloud) (10.0.1)
Requirement already satisfied: matplotlib in
/home/xbmu/.local/lib/python3.8/site-packages (from wordcloud) (3.7.3)
Requirement already satisfied: contourpy>=1.0.1 in
/home/xbmu/.local/lib/python3.8/site-packages (from matplotlib->wordcloud)
(1.1.1)
Requirement already satisfied: cycler>=0.10 in
/home/xbmu/.local/lib/python3.8/site-packages (from matplotlib->wordcloud)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/home/xbmu/.local/lib/python3.8/site-packages (from matplotlib->wordcloud)
(4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/xbmu/.local/lib/python3.8/site-packages (from matplotlib->wordcloud)
(1.4.5)
Requirement already satisfied: packaging>=20.0 in
/home/xbmu/.local/lib/python3.8/site-packages (from matplotlib->wordcloud)
(23.1)
Requirement already satisfied: pyparsing>=2.3.1 in
/home/xbmu/.local/lib/python3.8/site-packages (from matplotlib->wordcloud)
(3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/home/xbmu/.local/lib/python3.8/site-packages (from matplotlib->wordcloud)
(2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in
/home/xbmu/.local/lib/python3.8/site-packages (from matplotlib->wordcloud)
(6.1.0)
Requirement already satisfied: zipp>=3.1.0 in
/home/xbmu/.local/lib/python3.8/site-packages (from importlib-
resources>=3.2.0->matplotlib->wordcloud) (3.17.0)
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from
python-dateutil>=2.7->matplotlib->wordcloud) (1.14.0)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: nltk in /home/xbmu/.local/lib/python3.8/site-
packages (3.8.1)
Requirement already satisfied: click in /home/xbmu/.local/lib/python3.8/site-
packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /home/xbmu/.local/lib/python3.8/site-
packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in
/home/xbmu/.local/lib/python3.8/site-packages (from nltk) (2023.8.8)
Requirement already satisfied: tqdm in /home/xbmu/.local/lib/python3.8/site-
packages (from nltk) (4.66.1)
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: unidecode in
/home/xbmu/.local/lib/python3.8/site-packages (1.3.6)
```

```
[nltk_data] Downloading package brown to /home/xbmu/nltk_data…
[nltk_data]   Package brown is already up-to-date!
[nltk_data] Downloading package cess_esp to /home/xbmu/nltk_data…
[nltk_data]   Package cess_esp is already up-to-date!
[nltk_data] Downloading package stopwords to /home/xbmu/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[2]:
```python
# funciones auxiliares
def get_frequencies(vocabulary: Counter, n: int) -> list:
    return [_[1] for _ in vocabulary.most_common(n)]

def plot_frequencies(frequencies: list, title="Freq of words"):
    x = list(range(1, len(frequencies)+1))
    plt.plot(x, frequencies, "-v")
    plt.xlabel("Freq rank (r)")
    plt.ylabel("Freq (f)")
    plt.title(title)


def extract_words_from_sentence(sentence: str) -> list:
    return sub(r'[^\w\s\']', ' ', sentence).lower().split()

def preprocess_corpus(corpus):
    # Obtener la oración de L1,
    # quitar signos de puntuación y
    # obtiene la lista de palabras
    word_list_l1 = []
    word_list_l2 = []
    for row in corpus:
        word_list_l1.extend(extract_words_from_sentence(row[0]))
    # Obtener la oración de L1,
    # quitar signos de puntuación y
    # obtiene la lista de palabras
        word_list_l2.extend(extract_words_from_sentence(row[1]))
    return word_list_l1, word_list_l2


def extract_pos_tags(tagged_lists):
    """
    Extracts POS tags from a list of lists of tagged words.

    Args:
        tagged_lists (list of list of tuple): List of lists where each sublist␣
 ↪contains (word, POS) tuples.
```

```
    Returns:
        list: A list containing the POS tags extracted from each sublist.
    """
    return list(zip(*[[tup[1] for tup in sublist] for sublist in␣
 ↪tagged_lists]))[0]

def expand_to_characters(word_list):
    character_list = []

    for sublist in word_list:
        for word in sublist:
            characters = list(word)
            character_list.extend(characters)

    return character_list

def generate_ngrams(word_list, n=2):
    ngram_list = []

    for sublist in word_list:
        for word in sublist:
            if len(word) >= n:
                for i in range(len(word) - n + 1):
                    ngram = word[i:i+n]
                    ngram_list.append(ngram)

    return ngram_list

def generate_ngrams_simple_list(word_list, n=2):
    ngram_list = []

    for word in word_list:
        if len(word) >= n:
            for i in range(len(word) - n + 1):
                ngram = word[i:i+n]
                ngram_list.append(ngram)

    return ngram_list


def normalize_strings(strings):
    normalized_strings = []

    for s in strings:
            # Remove accents and convert to lowercase
            normalized_string = unidecode(s).lower()
```

```
            normalized_string = ''.join(char for char in normalized_string if␣
 ↪char not in string.punctuation)
            if normalized_string != '':
                normalized_strings.append(normalized_string)

    return normalized_strings
```

0.2 1. Etiquetas POS

[3]:
```
eng_tagged = brown.tagged_sents()
spa_tagged = cess.tagged_sents()
```

[4]:
```
eng_tags = Counter(extract_pos_tags(eng_tagged))
spa_tags = Counter(extract_pos_tags(spa_tagged))
```

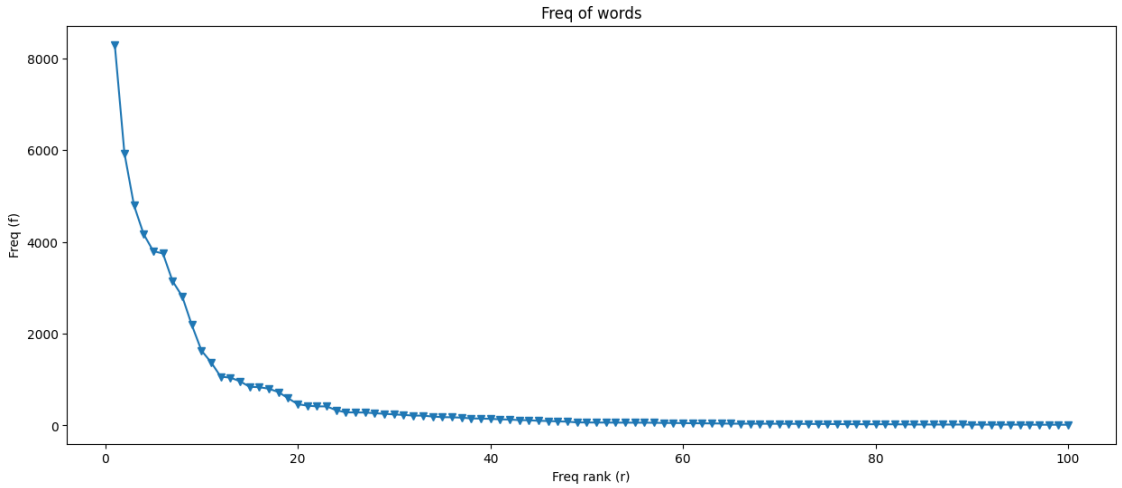
[5]:
```
eng_tags.most_common(10)
```

[5]:
```
[('AT', 8297),
 ('PPS', 5935),
 ('IN', 4792),
 ('``', 4168),
 ('RB', 3799),
 ('NP', 3748),
 ('PPSS', 3146),
 ('CC', 2815),
 ('CS', 2189),
 ('DT', 1633)]
```

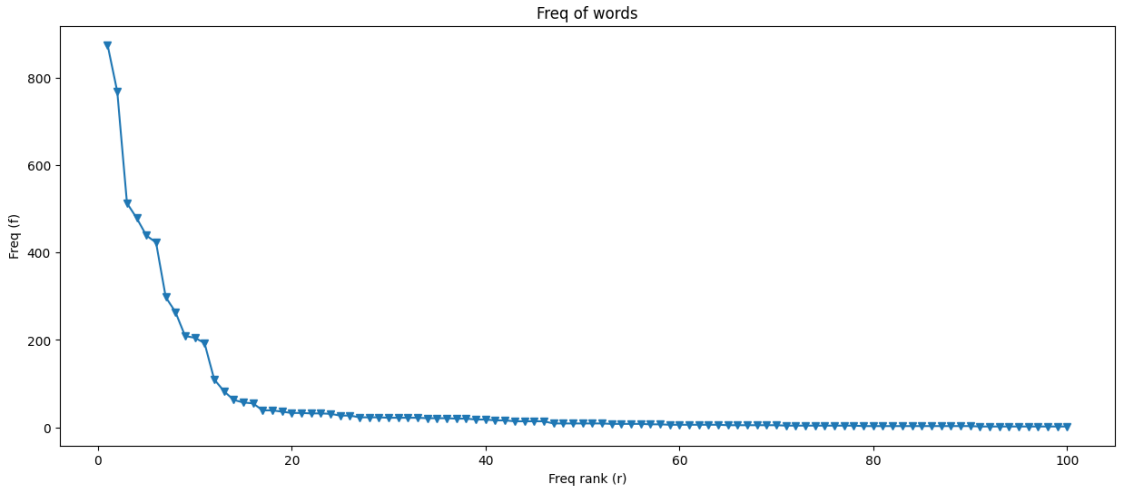
[6]:
```
spa_tags.most_common(10)
```

[6]:
```
[('sps00', 875),
 ('da0ms0', 768),
 ('rg', 513),
 ('da0fs0', 479),
 ('np0000p', 439),
 ('sn.e-SUJ', 423),
 ('cc', 298),
 ('Fe', 264),
 ('da0mp0', 209),
 ('Fg', 205)]
```

[7]:
```
frequencies = get_frequencies(eng_tags, 100)
plot_frequencies(frequencies)
```



[8]:
```
frequencies = get_frequencies(spa_tags, 100)
plot_frequencies(frequencies)
```



0.3 2. Caractéres

[9]:
```
brown.sents()
```

[9]:
```
[['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an',
'investigation', 'of', "Atlanta's", 'recent', 'primary', 'election', 'produced',
'``', 'no', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'place',
'.'], ['The', 'jury', 'further', 'said', 'in', 'term-end', 'presentments',
'that', 'the', 'City', 'Executive', 'Committee', ',', 'which', 'had', 'over-
all', 'charge', 'of', 'the', 'election', ',', '``', 'deserves', 'the', 'praise',
```

```
'and', 'thanks', 'of', 'the', 'City', 'of', 'Atlanta', "''", 'for', 'the',
'manner', 'in', 'which', 'the', 'election', 'was', 'conducted', '.'], …]
```

[10]:
```
eng_chars = Counter(expand_to_characters(brown.sents()))
spa_chars = Counter(expand_to_characters(cess.sents()))
spanish_words_na, nahuatl_words = preprocess_corpus(axolotl)
spanish_words_oto, otomi_words = preprocess_corpus(tsunkua)
nahuatl_chars = Counter(expand_to_characters(nahuatl_words))
otomi_chars = Counter(expand_to_characters(otomi_words))
```

[11]:
```
nahuatl_chars.most_common(10)
```

[11]:
```
[('a', 240008),
 ('i', 218038),
 ('n', 162612),
 ('t', 149615),
 ('o', 127941),
 ('c', 125389),
 ('l', 122619),
 ('u', 120714),
 ('e', 105715),
 ('h', 94779)]
```

[12]:
```
otomi_chars.most_common(10)
```

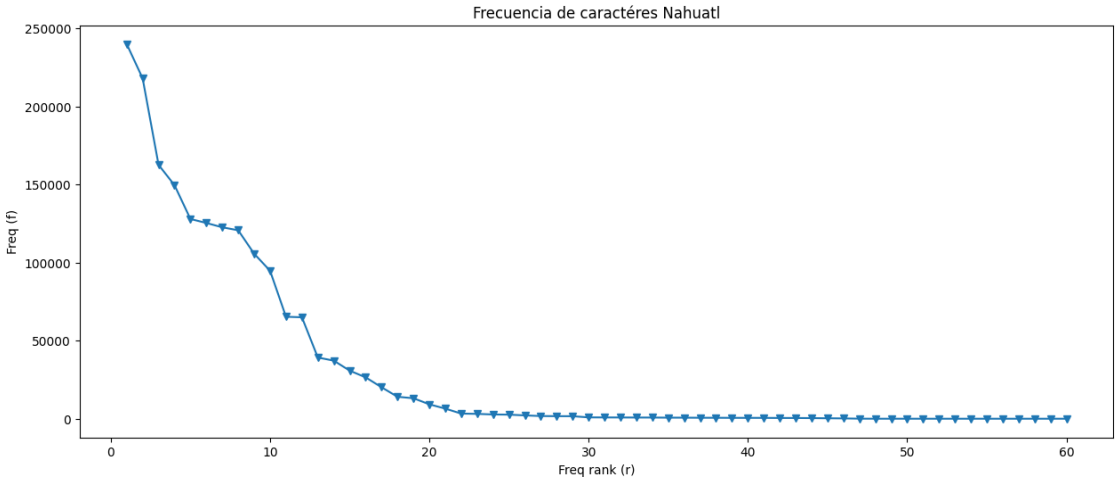
[12]:
```
[('a', 34672),
 ('i', 23253),
 ('n', 19120),
 ('t', 14246),
 ('u', 13753),
 ('e', 13705),
 ("'", 13176),
 ('h', 13068),
 ('o', 11874),
 ('m', 9672)]
```

[13]:
```
spa_chars.most_common(10)
```

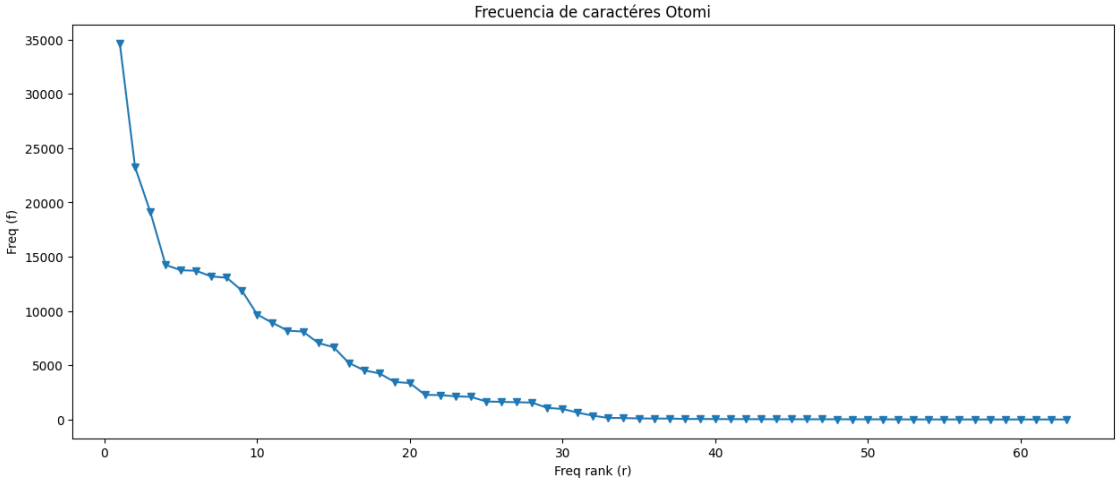
[13]:
```
[('e', 107336),
 ('a', 99691),
 ('o', 71002),
 ('s', 59962),
 ('n', 58840),
 ('r', 55694),
 ('i', 53313),
 ('l', 47821),
 ('d', 42892),
```

```
 ('t', 37305)]
```

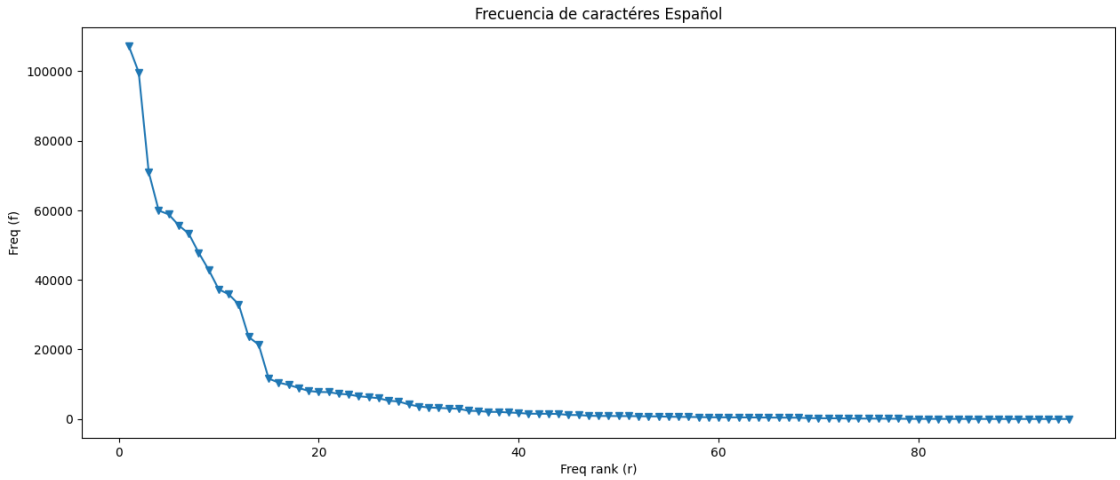
[14]:
```
most_common_count_chars = 100
frequencies = get_frequencies(nahuatl_chars, most_common_count_chars)
plot_frequencies(frequencies,"Frecuencia de caractéres Nahuatl")
```



[15]:
```
frequencies = get_frequencies(otomi_chars, most_common_count_chars)
plot_frequencies(frequencies,"Frecuencia de caractéres Otomi")
```



[16]:
```
frequencies = get_frequencies(spa_chars, most_common_count_chars)
plot_frequencies(frequencies,"Frecuencia de caractéres Español")
```



0.4 3. n-gramas

[17]:
```
eng_gram = Counter(normalize_strings(generate_ngrams(brown.sents(), n=2)))
spa_gram = Counter(normalize_strings(generate_ngrams(cess.sents(), n=2)))
nahuatl_gram =␣
 ↪Counter(normalize_strings(generate_ngrams_simple_list(nahuatl_words)))
otomi_gram =␣
 ↪Counter(normalize_strings(generate_ngrams_simple_list(otomi_words)))
```

[18]:
```
nahuatl_gram.most_common(10)
```

[18]:
```
[('tl', 53811),
 ('an', 50850),
 ('ca', 44109),
 ('ui', 41846),
 ('hu', 39456),
 ('qu', 39140),
 ('in', 38382),
 ('ua', 36644),
 ('la', 34170),
 ('ic', 28553)]
```

[19]:
```
otomi_gram.most_common(10)
```

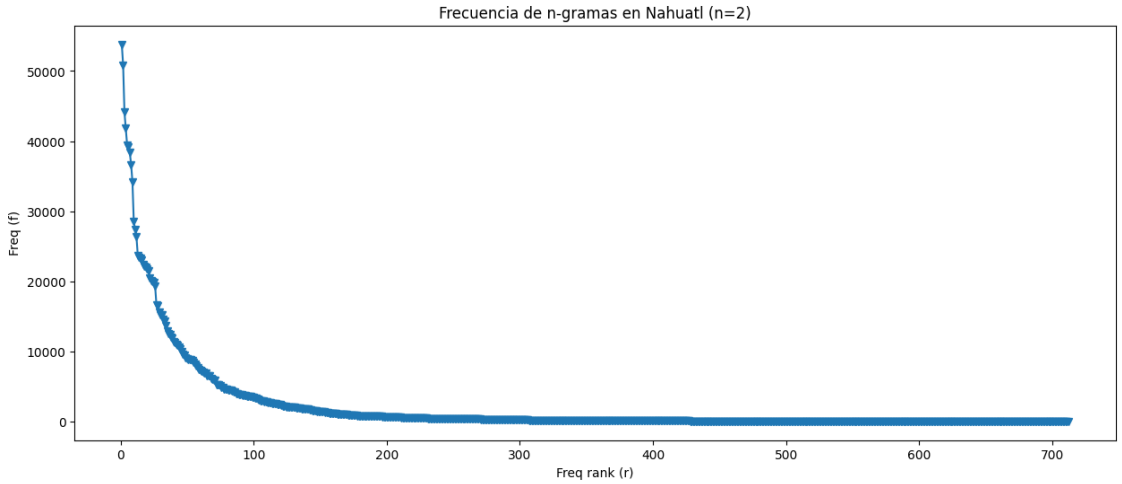
[19]:
```
[('ra', 6668),
 ('ya', 5291),
 ('a', 5080),
 ('bi', 3620),
 ('ts', 3493),
 ('ma', 3141),
```

```
 ('nu', 3128),
 ('di', 2789),
 ('e', 2770),
 ('da', 2727)]
```

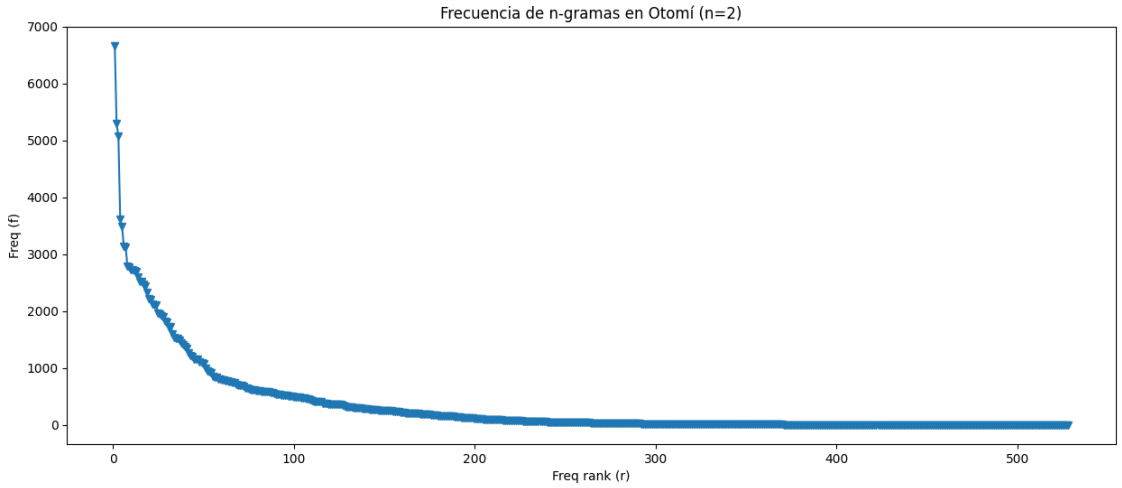
[20]:
```
spa_gram.most_common(10)
```

[20]:
```
[('de', 21237),
 ('en', 18440),
 ('es', 16502),
 ('la', 13499),
 ('on', 13253),
 ('os', 12918),
 ('er', 12113),
 ('ra', 11707),
 ('el', 11637),
 ('an', 11126)]
```

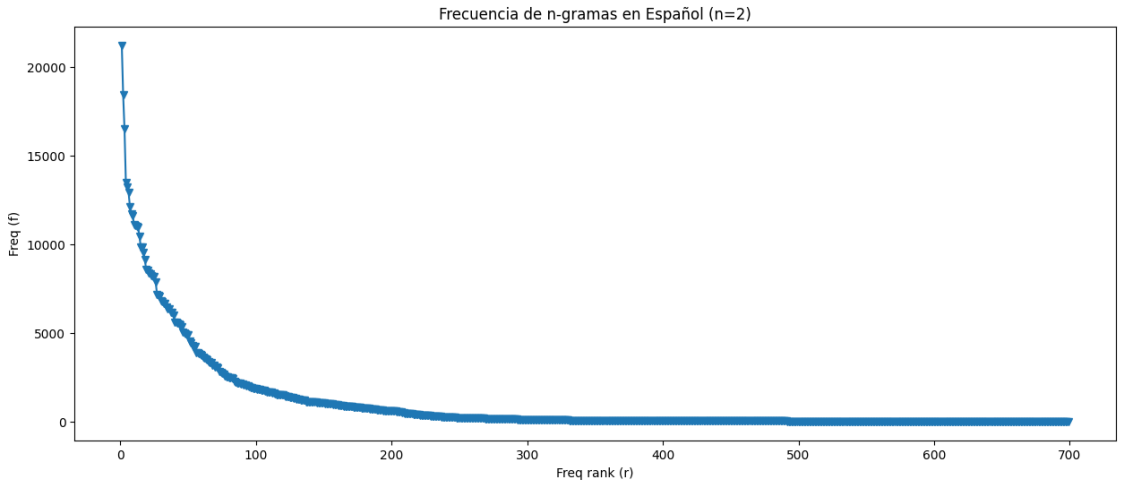
[21]:
```
frequencies = get_frequencies(nahuatl_gram, 1000)
plot_frequencies(frequencies,"Frecuencia de n-gramas en Nahuatl (n=2)")
```



[22]:
```
frequencies = get_frequencies(otomi_gram, 1000)
plot_frequencies(frequencies,"Frecuencia de n-gramas en Otomí (n=2)")
```



[23]:
```
frequencies = get_frequencies(spa_gram, 1000)
plot_frequencies(frequencies,"Frecuencia de n-gramas en Español (n=2)")
```



0.5 4. stopwords

[24]:
```
stopwords_eng = stopwords.words('english')
```

[25]:
```
len(stopwords_eng)
```

[25]: 179

[26]:
```
stopwords_eng[1:10]
```

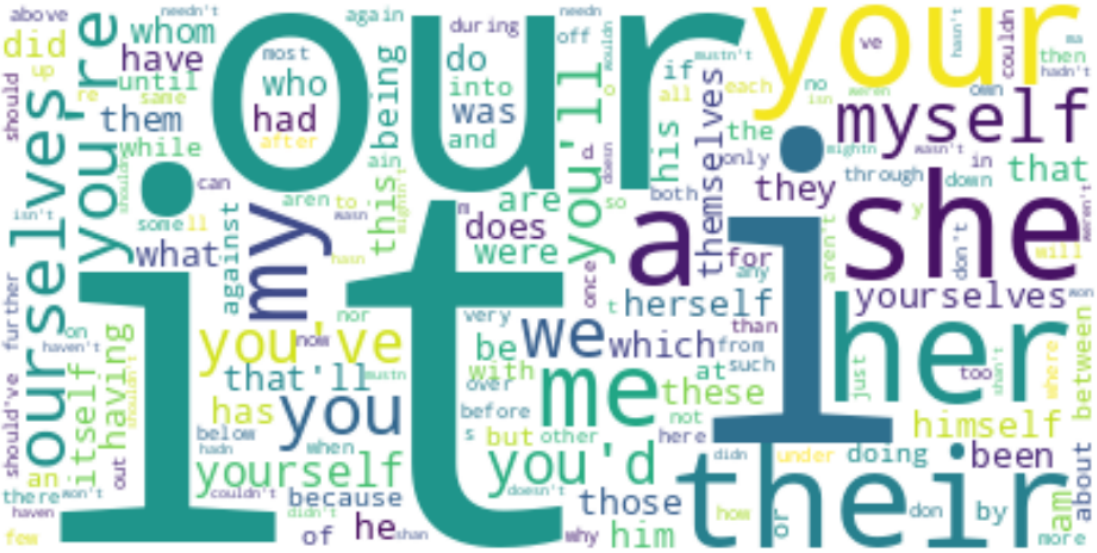
[26]:
```
['me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

[27]:
```
generated_stopwords = Counter( normalize_strings(brown.words()) ).
↪most_common(len(stopwords_eng))
generated_stopword_list = [t[0] for t in generated_stopwords]
sorted(generated_stopword_list)[1:10]
```

[27]:
```
['a', 'about', 'af', 'after', 'again', 'against', 'all', 'also', 'american']
```

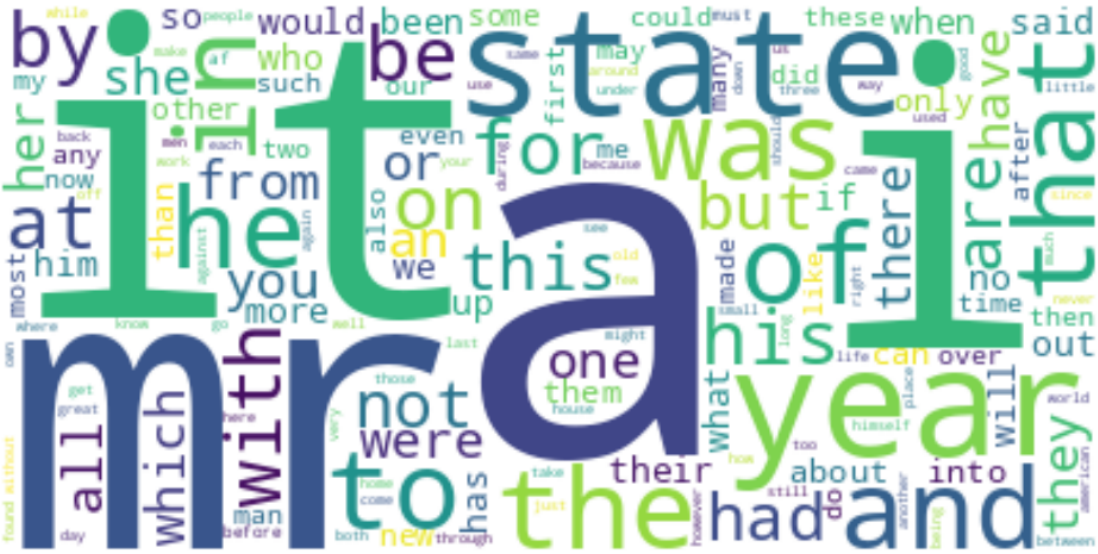
0.5.1 Nube de Stopwords de NLTK

[28]:
```
word_cloud = WordCloud(collocations = True, background_color = 'white',␣
↪include_numbers = False, stopwords = [""]).generate(" ".join(stopwords_eng))
plt.imshow(word_cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



0.5.2 Nube de Stopwords generadas con Zipf

[29]:
```
generated_word_cloud = WordCloud(collocations = True, background_color =␣
↪'white', include_numbers = False, stopwords = [""]).generate(" ".
↪join(generated_stopword_list))
plt.imshow(generated_word_cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



0.5.3 ¿Obtenemos el mismo resultado? Si o no y ¿Porqué?

No, en nuestras stopwords generadas vía Zipf existen palabras que a pesar de ser muy comunes sí agregan información, tal es el caso de 'state', 'year, 'first', 'people', en cambio, otras palabras que carecen de sentido por sí mismas o que agregan poca o nula información no aparecen en sus variantes menos comunes como 'mightn', "mightn't"

[ ]: