Carga de Paquetes

In [2]:
```
pip install newspaper3k
```

```
Collecting newspaper3k
  Downloading newspaper3k-0.2.8-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: beautifulsoup4>=4.4.1 in c:\users\pc\anaconda3\lib\si
te-packages (from newspaper3k) (4.12.2)
Requirement already satisfied: Pillow>=3.3.0 in c:\users\pc\anaconda3\lib\site-packa
ges (from newspaper3k) (10.2.0)
Requirement already satisfied: PyYAML>=3.11 in c:\users\pc\anaconda3\lib\site-packag
es (from newspaper3k) (6.0.1)
Requirement already satisfied: cssselect>=0.9.2 in c:\users\pc\anaconda3\lib\site-pa
ckages (from newspaper3k) (1.2.0)
Requirement already satisfied: lxml>=3.6.0 in c:\users\pc\anaconda3\lib\site-package
s (from newspaper3k) (4.9.3)
Requirement already satisfied: nltk>=3.2.1 in c:\users\pc\anaconda3\lib\site-package
s (from newspaper3k) (3.8.1)
Requirement already satisfied: requests>=2.10.0 in c:\users\pc\anaconda3\lib\site-pa
ckages (from newspaper3k) (2.31.0)
Collecting feedparser>=5.2.1 (from newspaper3k)
  Downloading feedparser-6.0.11-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: tldextract>=2.0.1 in c:\users\pc\anaconda3\lib\site-p
ackages (from newspaper3k) (3.2.0)
Collecting feedfinder2>=0.0.4 (from newspaper3k)
  Downloading feedfinder2-0.0.4.tar.gz (3.3 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting jieba3k>=0.35.1 (from newspaper3k)
  Downloading jieba3k-0.35.1.zip (7.4 MB)
     ---------------------------------------- 0.0/7.4 MB ? eta -:--:--
     ---------------------------------------- 0.0/7.4 MB ? eta -:--:--
     ---------------------------------------- 0.0/7.4 MB ? eta -:--:--
     ---------------------------------------- 0.0/7.4 MB 281.8 kB/s eta 0:00:27
     ---------------------------------------- 0.1/7.4 MB 585.1 kB/s eta 0:00:13
     - -------------------------------------- 0.3/7.4 MB 1.3 MB/s eta 0:00:06
     -- ------------------------------------- 0.4/7.4 MB 1.8 MB/s eta 0:00:05
     ---- ----------------------------------- 0.8/7.4 MB 2.7 MB/s eta 0:00:03
     ------ --------------------------------- 1.1/7.4 MB 3.6 MB/s eta 0:00:02
     ------ --------------------------------- 1.1/7.4 MB 3.6 MB/s eta 0:00:02
     --------- ------------------------------ 1.7/7.4 MB 4.0 MB/s eta 0:00:02
     ------------ --------------------------- 2.4/7.4 MB 5.2 MB/s eta 0:00:01
     ------------ --------------------------- 2.4/7.4 MB 4.7 MB/s eta 0:00:02
     --------------- ------------------------ 2.8/7.4 MB 5.1 MB/s eta 0:00:01
     ------------------- -------------------- 3.6/7.4 MB 5.8 MB/s eta 0:00:01
     ------------------- -------------------- 3.6/7.4 MB 5.8 MB/s eta 0:00:01
     ---------------------- ----------------- 4.3/7.4 MB 6.0 MB/s eta 0:00:01
     ------------------------ --------------- 4.6/7.4 MB 6.1 MB/s eta 0:00:01
     ------------------------- -------------- 4.9/7.4 MB 6.1 MB/s eta 0:00:01
     --------------------------- ------------ 5.3/7.4 MB 6.2 MB/s eta 0:00:01
     ----------------------------- ---------- 5.6/7.4 MB 6.3 MB/s eta 0:00:01
     ------------------------------- -------- 6.0/7.4 MB 6.4 MB/s eta 0:00:01
     --------------------------------- ------ 6.3/7.4 MB 6.4 MB/s eta 0:00:01
     ---------------------------------- ----- 6.6/7.4 MB 6.4 MB/s eta 0:00:01
     ------------------------------------ --- 7.0/7.4 MB 6.5 MB/s eta 0:00:01
     -------------------------------------  7.3/7.4 MB 6.5 MB/s eta 0:00:01
     -------------------------------------  7.4/7.4 MB 6.4 MB/s eta 0:00:01
     ---------------------------------------- 7.4/7.4 MB 6.1 MB/s eta 0:00:00
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
```

```
Requirement already satisfied: python-dateutil>=2.5.3 in c:\users\pc\anaconda3\lib\s
ite-packages (from newspaper3k) (2.8.2)
Collecting tinysegmenter==0.3 (from newspaper3k)
  Downloading tinysegmenter-0.3.tar.gz (16 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: soupsieve>1.2 in c:\users\pc\anaconda3\lib\site-packa
ges (from beautifulsoup4>=4.4.1->newspaper3k) (2.5)
Requirement already satisfied: six in c:\users\pc\anaconda3\lib\site-packages (from
feedfinder2>=0.0.4->newspaper3k) (1.16.0)
Collecting sgmllib3k (from feedparser>=5.2.1->newspaper3k)
  Downloading sgmllib3k-1.0.0.tar.gz (5.8 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: click in c:\users\pc\anaconda3\lib\site-packages (fro
m nltk>=3.2.1->newspaper3k) (8.1.7)
Requirement already satisfied: joblib in c:\users\pc\anaconda3\lib\site-packages (fr
om nltk>=3.2.1->newspaper3k) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\pc\anaconda3\lib\site-pac
kages (from nltk>=3.2.1->newspaper3k) (2023.10.3)
Requirement already satisfied: tqdm in c:\users\pc\anaconda3\lib\site-packages (from
nltk>=3.2.1->newspaper3k) (4.65.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\pc\anaconda3\lib
\site-packages (from requests>=2.10.0->newspaper3k) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\pc\anaconda3\lib\site-packag
es (from requests>=2.10.0->newspaper3k) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\pc\anaconda3\lib\site-
packages (from requests>=2.10.0->newspaper3k) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\pc\anaconda3\lib\site-
packages (from requests>=2.10.0->newspaper3k) (2024.2.2)
Requirement already satisfied: requests-file>=1.4 in c:\users\pc\anaconda3\lib\site-
packages (from tldextract>=2.0.1->newspaper3k) (1.5.1)
Requirement already satisfied: filelock>=3.0.8 in c:\users\pc\anaconda3\lib\site-pac
kages (from tldextract>=2.0.1->newspaper3k) (3.13.1)
Requirement already satisfied: colorama in c:\users\pc\anaconda3\lib\site-packages
(from click->nltk>=3.2.1->newspaper3k) (0.4.6)
Downloading newspaper3k-0.2.8-py3-none-any.whl (211 kB)
   ---------------------------------------- 0.0/211.1 kB ? eta -:--:--
   ---------------------------------------- 211.1/211.1 kB 6.5 MB/s eta 0:00:00
Downloading feedparser-6.0.11-py3-none-any.whl (81 kB)
   ---------------------------------------- 0.0/81.3 kB ? eta -:--:--
   ---------------------------------------- 81.3/81.3 kB 4.4 MB/s eta 0:00:00
Building wheels for collected packages: tinysegmenter, feedfinder2, jieba3k, sgmllib
3k
  Building wheel for tinysegmenter (setup.py): started
  Building wheel for tinysegmenter (setup.py): finished with status 'done'
  Created wheel for tinysegmenter: filename=tinysegmenter-0.3-py3-none-any.whl size=
13567 sha256=fab9ba3cadf61ac878920400113b4e34c0f97051e790c48c9c85ced144ddbf9e
  Stored in directory: c:\users\pc\appdata\local\pip\cache\wheels\fc\ab\f8\cce3a9ae6
d828bd346be695f7ff54612cd22b7cbd7208d68f3
  Building wheel for feedfinder2 (setup.py): started
  Building wheel for feedfinder2 (setup.py): finished with status 'done'
  Created wheel for feedfinder2: filename=feedfinder2-0.0.4-py3-none-any.whl size=33
57 sha256=f3bda1d472be6a9b76cacf74897316726ea9cda52d6bbda736596d2061b8e642
  Stored in directory: c:\users\pc\appdata\local\pip\cache\wheels\80\d5\72\9cd9eccc8
19636436c6a6e59c22a0fb1ec167beef141f56491
```

```
  Building wheel for jieba3k (setup.py): started
  Building wheel for jieba3k (setup.py): finished with status 'done'
  Created wheel for jieba3k: filename=jieba3k-0.35.1-py3-none-any.whl size=7398387 s
ha256=f96d491450894ddb531fee3dd3b260a4ddaddc6fe70f796ab5751c8d533abd1c
  Stored in directory: c:\users\pc\appdata\local\pip\cache\wheels\3a\a1\46\8e68055c1
713f9c4598774c15ad0541f26d5425ee7423b6493
  Building wheel for sgmllib3k (setup.py): started
  Building wheel for sgmllib3k (setup.py): finished with status 'done'
  Created wheel for sgmllib3k: filename=sgmllib3k-1.0.0-py3-none-any.whl size=6061 s
ha256=ce6de7aea2d278b90ec84528b8f0120638f4820a3181a9f9282fef2ab9449ae8
  Stored in directory: c:\users\pc\appdata\local\pip\cache\wheels\3b\25\2a\105d6a15d
f6914f4d15047691c6c28f9052cc1173e40285d03
Successfully built tinysegmenter feedfinder2 jieba3k sgmllib3k
Installing collected packages: tinysegmenter, sgmllib3k, jieba3k, feedparser, feedfi
nder2, newspaper3k
Successfully installed feedfinder2-0.0.4 feedparser-6.0.11 jieba3k-0.35.1 newspaper3
k-0.2.8 sgmllib3k-1.0.0 tinysegmenter-0.3
Note: you may need to restart the kernel to use updated packages.
```

Carga de Librerias

In [64]:
```
import newspaper
import json
import gutenbergpy.textget
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus   import stopwords
import matplotlib.pyplot as plt
import seaborn as sns
#Derivación regresiva textos en español
from nltk.stem import SnowballStemmer
from collections import Counter
import matplotlib.pyplot as plt
from nltk.util import ngrams
import emoji
import networkx as nx
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
```

Obtención de datos

In [65]:
```
# Carga de noticias del 2025, sobre el aguacate que se usa en el Super BOWL
url= 'https://www.milenio.com/negocios/exportacion-aguacate-eu-super-bowl-110-mil-t

articulo25 = newspaper.Article(url=url, language='es')
articulo25.download()
articulo25.parse()

articulo25 ={
    "Titulo": str(articulo25.title),
    "Texto": str(articulo25.text),
    "Autor": articulo25.authors,
```

```
    "Fecha_publicacion": str(articulo25.publish_date),
    "Imagen": str(articulo25.top_image),
    "Videos": articulo25.movies,
    "keywords": articulo25.keywords,
    "Resumen": str(articulo25.summary)
}

print(articulo25["Titulo"] \
     + "\n\t\t" + articulo25["Fecha_publicacion"] \
     + "\n\n"\
   #  + "\n" + articulo["Texto"]\
     + "\n\n")
```

```
Exportación de aguacate a EU por Super Bowl-110 mil toneladas: Apeam
                2025-01-14 20:30:00
```

In [66]:
```
# Carga de noticias del 2024, sobre el aguacate que se usa en el Super BOWL
url= 'https://www.milenio.com/negocios/financial-times/super-bowl-dispara-demanda-d

articulo24 = newspaper.Article(url=url, language='es')
articulo24.download()
articulo24.parse()

articulo24 ={
    "Titulo": str(articulo24.title),
    "Texto": str(articulo24.text),
    "Autor": articulo24.authors,
    "Fecha_publicacion": str(articulo24.publish_date),
    "Imagen": str(articulo24.top_image),
    "Videos": articulo24.movies,
    "keywords": articulo24.keywords,
    "Resumen": str(articulo24.summary)
}

print(articulo24["Titulo"] \
     + "\n\t\t" + articulo24["Fecha_publicacion"] \
     + "\n\n"\
   #  + "\n" + articulo24["Texto"]\
     + "\n\n")
```

```
Super Bowl dispara demanda del aguacate
                2024-09-02 13:43:00
```

Tokenizar Texto

In [67]:
```
texto25 = articulo25["Texto"]
```

```
texto24 = articulo24["Texto"]
```

In [68]:
```
palabras_24 = word_tokenize(texto24,"spanish")
palabrasfiltradas24=[word.lower() for word in palabras_24 if word.isalpha()] # Remo
#print(palabrasfiltradas24)
```

In [69]:
```
palabras_25 = word_tokenize(texto25,"spanish")
palabrasfiltradas25=[word.lower() for word in palabras_25 if word.isalpha()] # Remo
#print(palabrasfiltradas25)
```

In [70]:
```
#Verificar frecuencia de palabras
freq24 = nltk.FreqDist(palabrasfiltradas24)
for key,val in freq24.items():
    print (str(key) + ':' + str(val))
```

el:38
aguacate:24
mexicano:3
está:3
listo:1
para:15
robarse:1
show:1
en:38
super:5
bowl:5
lviii:2
entre:2
los:20
chiefs:2
de:84
kansas:2
city:2
y:24
san:2
francisco:2
con:10
proyecciones:1
que:25
indican:1
este:7
año:7
se:22
batirán:1
récords:1
ventas:3
estados:5
unidos:4
eu:10
evento:3
tradicionalmente:1
dispara:1
la:32
demanda:4
del:18
promete:1
ser:1
especialmente:1
lucrativo:1
exportadores:4
mexicanos:3
a:20
pesar:1
desafíos:1
precios:1
tipo:1
cambio:1
las:6
estadísticas:1
hablan:1
por:8

```
sí:1
solas:1
esta:5
temporada:6
méxico:9
provee:1
aproximadamente:1
ciento:3
aguacates:3
consumen:2
prepara:1
superar:1
cifras:2
exportación:5
años:2
anteriores:1
diciembre:1
país:4
ya:4
había:1
logrado:2
un:6
hito:1
al:3
exportar:3
acumulado:2
toneladas:8
superando:1
récord:1
previo:1
último:1
mes:1
fue:1
mil:10
incremento:1
es:10
solo:4
una:3
muestra:1
potencial:1
espera:1
concretar:1
dice:1
armando:1
lópez:1
orduña:1
ceo:1
asociación:3
productores:9
empacadores:1
apeam:3
integrada:1
industria:2
aguacatera:1
luego:1
trabajo:1
```

```
ha:4
convertido:2
representante:1
más:4
grande:2
siendo:1
destino:2
principal:3
fundó:1
cinco:1
empacadoras:2
huertas:1
exportaron:3
seis:1
durante:4
cuatro:1
meses:2
ese:1
hoy:3
somos:1
fortalece:1
todo:1
cadena:1
valor:2
producción:3
forma:1
positiva:1
gracias:3
ello:1
mejorado:1
calidad:4
vida:1
quienes:1
integran:1
importante:3
además:2
disminuir:1
migración:1
razón:1
producto:4
agropecuario:1
considerado:1
oro:1
verde:1
creció:1
hemos:2
promovido:1
fruto:3
otros:1
países:4
bajo:1
marca:3
avocados:4
from:4
mexico:4
haciendo:1
```

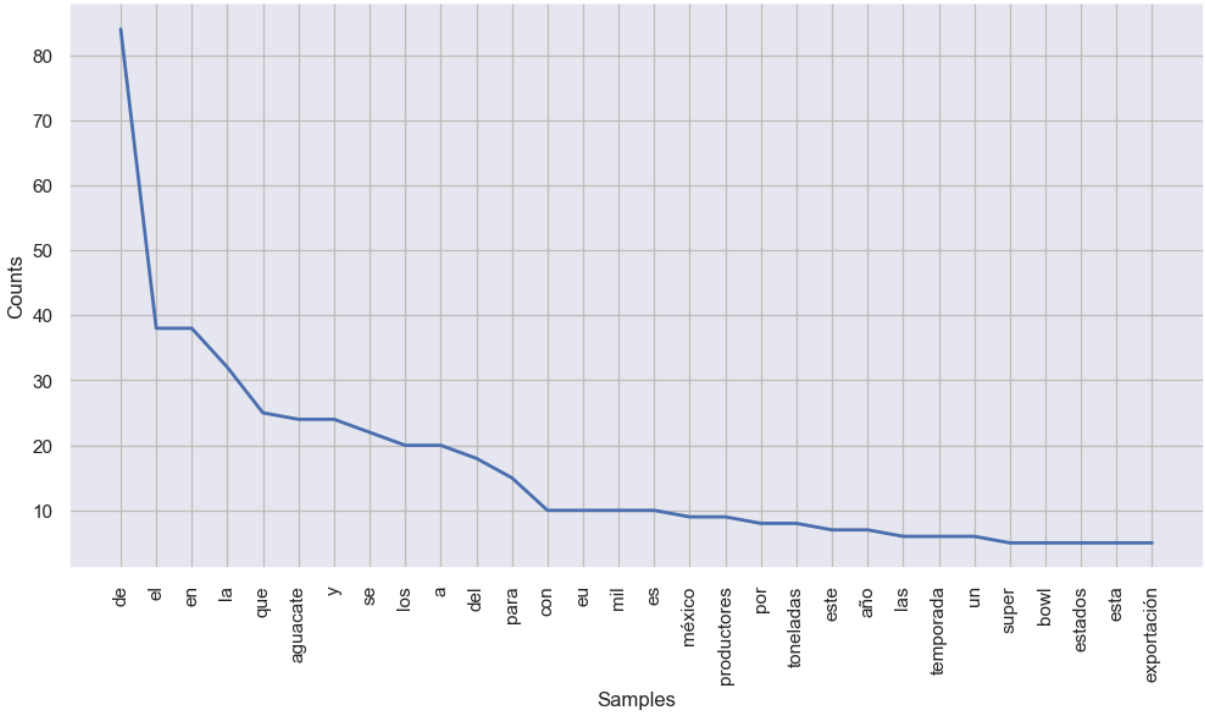
ícono:1
internacional:1
campaña:3
aumentó:1
su:1
posicionamiento:1
mundo:1
tras:1
colocando:1
como:4
uno:3
principales:2
nivel:2
mundial:2
pasar:1
exportadas:1
principios:1
llegamos:1
seguido:1
canadá:3
japón:3
francia:1
salvador:1
españa:1
honduras:1
holanda:1
china:4
corea:4
sur:3
qatar:1
cuanto:1
consumo:2
estadounidense:3
pasó:1
gramos:1
per:1
cápita:1
kilogramos:1
persona:1
día:1
gran:1
medida:1
esfuerzan:1
constantemente:1
sembrar:1
cosechar:1
limpio:1
importancia:1
nace:1
finalidad:1
llevar:1
número:2
vecino:1
según:2
departamento:1
agricultura:1

cada:2
mercado:1
provienen:1
éxito:1
cosechando:1
haber:1
diseñado:1
estrategias:1
correctas:1
decir:1
dirigir:1
campañas:1
publicitarias:1
específicas:1
tanto:1
público:2
latino:1
volverá:1
aparecer:1
comercial:1
segundos:1
mostrará:1
transmitirá:1
antes:1
arranque:1
segundo:1
cuarto:1
partido:2
destinamos:1
presupuesto:1
marketing:1
porque:2
si:1
no:5
lo:2
hacemos:1
posicionan:1
ni:1
venden:1
cabe:1
destacar:1
dura:1
después:1
viene:1
cuaresma:1
mayo:1
cual:1
fecha:1
muy:1
incrementa:1
beneficios:1
centran:1
también:2
llegan:1
será:1
único:1

fresco:1
presente:1
aprobación:1
tan:1
guacamole:2
estrella:1
promedio:2
consume:1
cerca:2
millones:3
cascos:1
futbol:1
deportivo:1
abastecer:1
preparan:1
anticipación:2
incluso:1
tienen:2
apartada:1
o:2
vendida:1
considerar:1
buscan:1
sea:1
alta:1
mediante:1
intervención:1
servicio:1
nacional:1
sanidad:1
inocuidad:1
valida:1
estándares:1
ganarán:1
prevé:1
exportaciones:1
ronde:1
pesos:1
superior:1
casi:2
doble:1
alcanzado:1
pasado:1
perspectiva:1
crecimiento:2
futuro:1
nuestras:1
estimaciones:1
esperamos:2
hacia:1
detone:1
producir:1
venta:2
algunos:1
europa:1
nuestros:1

```
mercados:3
sino:1
tenemos:1
intención:1
abrir:1
espacios:1
india:1
chile:1
todos:1
donde:1
tocado:1
puertas:1
hay:2
buena:1
respuesta:1
mucho:1
interés:1
tiene:1
les:1
interesa:1
satisfacer:1
estos:1
crecieron:1
hectáreas:2
plantación:1
cuáles:1
cuentan:1
certificación:1
están:1
proceso:1
obtenerlo:1
djr:1
```

In [71]:
```python
# Configuración de Seaborn
sns.set()

# Crear la figura
plt.figure(figsize=(12, 6))
freq24.plot(30, cumulative=False)

plt.close()
```



In [72]:
```
#Verificar frecuencia de palabras
freq25 = nltk.FreqDist(palabrasfiltradas25)
for key,val in freq25.items():
    print (str(key) + ':' + str(val))
```

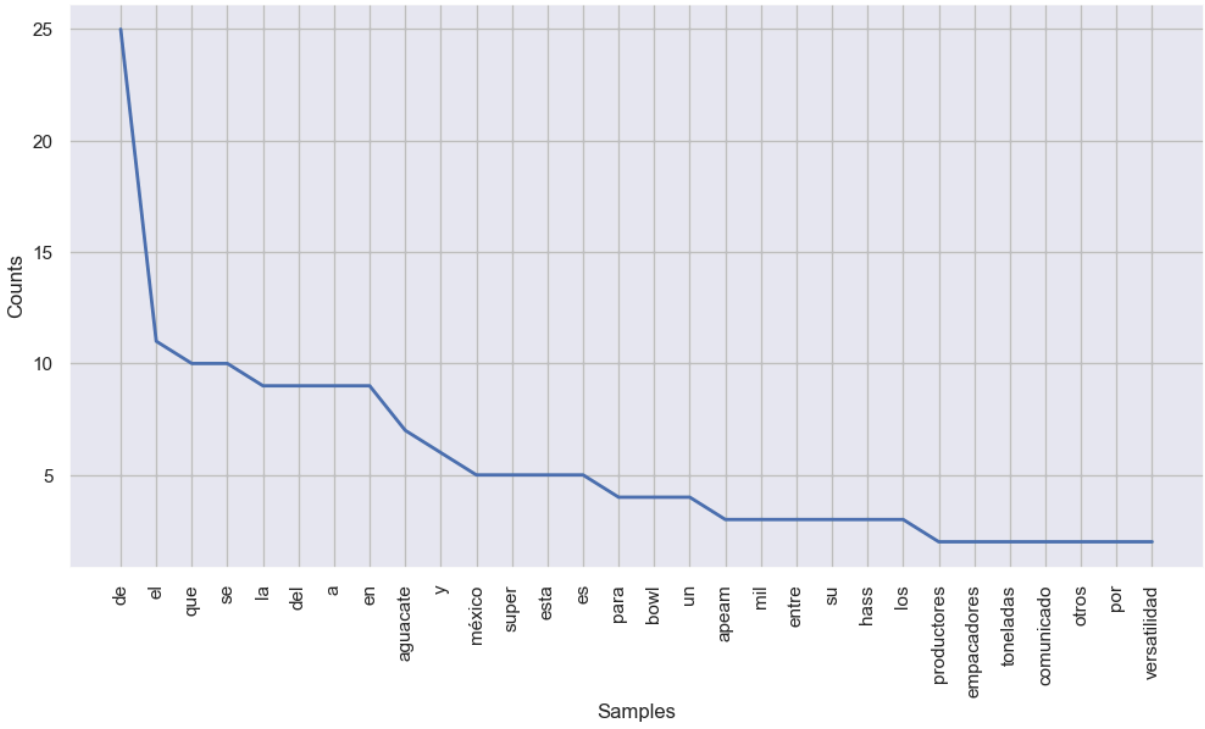
```
la:9
asociación:1
de:25
productores:2
y:6
empacadores:2
exportadores:1
aguacate:7
méxico:5
apeam:3
estimó:1
que:10
para:4
edición:1
lix:1
del:9
super:5
bowl:4
a:9
celebrarse:1
el:11
próximo:1
febrero:1
se:10
exportarán:1
mil:3
toneladas:2
oro:1
verde:1
través:1
un:4
comunicado:2
detalló:1
esta:5
cifra:1
es:5
similar:1
año:1
pasado:1
exportación:1
reflejo:1
arduo:1
trabajo:1
adaptación:1
dedicación:1
miles:1
mexicanos:1
entre:3
ingenieros:1
agrónomos:1
técnicos:1
campo:1
supervisores:1
seleccionadoras:1
frutas:1
otros:2
```

```
destacó:1
en:9
reconocido:1
por:2
su:3
inigualable:1
sabor:1
textura:1
cremosa:1
versatilidad:2
hass:3
ha:2
convertido:1
elemento:1
esencial:1
las:1
reuniones:1
especialmente:1
como:2
protagonista:1
famoso:1
guacamole:2
lee:1
nota:1
informativa:1
precisó:1
mayor:1
productor:1
mundial:1
aguacates:2
lo:2
este:2
evento:1
conlleva:1
impacto:1
economía:1
mexicana:1
estadunidense:2
sentido:1
señaló:1
fruta:2
elige:1
bajo:1
estrictas:1
medidas:1
fitosanidad:1
inocuidad:1
calidad:1
fin:1
llevar:1
mejor:1
los:3
espectadores:1
tazón:1
te:2
recomendamos:2
```

canasta:1
básica:1
puebla:1
así:1
incrementado:1
costo:1
años:1
negocios:2
asimismo:1
resaltó:1
estados:1
unidos:1
principal:1
consumidor:1
siendo:1
ingesta:1
per:1
cápita:1
kilogramos:1
sin:2
embargo:1
actualmente:1
exportan:1
más:2
países:1
todo:1
mundo:1
qué:1
favorito:1
acuerdo:1
con:2
tiene:1
una:1
culinaria:1
debido:1
platillos:1
salsas:1
ensaladas:1
también:1
mencionó:1
variedad:1
ofrece:1
grasas:1
saludables:1
fibra:1
vitaminas:1
antioxidantes:1
permiten:1
disfrutar:1
culpa:1
finalmente:1
expuso:1
temporada:1
firma:1
avocados:1
from:1

```
mexico:1
asoció:1
icónico:1
rob:1
gronkowski:1
exjugador:1
profesional:1
fútbol:1
americano:1
apoyar:1
compradores:1
prepararse:1
sus:1
fiestas:1
aws:1
probará:1
nuevas:1
tecnologías:1
sumarán:1
mdd:1
al:1
pib:1
kl:1
```

In [73]:
```
# Configuración de Seaborn
sns.set()

# Crear la figura
plt.figure(figsize=(12, 6))
freq25.plot(30, cumulative=False)

plt.close()
```



Eliminación de Palabras Vacías (Stopwords)

In [74]:
```
limpia_palabras24 = palabrasfiltradas24[:]

for palabra in palabrasfiltradas24:

    if palabra in stopwords.words('spanish'):

        limpia_palabras24.remove(palabra)
```

In [75]:
```
#Verificar frecuencia de palabras
freq_limpia24 = nltk.FreqDist(limpia_palabras24)
for key,val in freq_limpia24.items():
    print (str(key) + ':' + str(val))
```

aguacate:24
mexicano:3
listo:1
robarse:1
show:1
super:5
bowl:5
lviii:2
chiefs:2
kansas:2
city:2
san:2
francisco:2
proyecciones:1
indican:1
año:7
batirán:1
récords:1
ventas:3
unidos:4
eu:10
evento:3
tradicionalmente:1
dispara:1
demanda:4
promete:1
ser:1
especialmente:1
lucrativo:1
exportadores:4
mexicanos:3
pesar:1
desafíos:1
precios:1
tipo:1
cambio:1
estadísticas:1
hablan:1
solas:1
temporada:6
méxico:9
provee:1
aproximadamente:1
ciento:3
aguacates:3
consumen:2
prepara:1
superar:1
cifras:2
exportación:5
años:2
anteriores:1
diciembre:1
país:4
logrado:2
hito:1

exportar:3
acumulado:2
toneladas:8
superando:1
récord:1
previo:1
último:1
mes:1
mil:10
incremento:1
solo:4
muestra:1
potencial:1
espera:1
concretar:1
dice:1
armando:1
lópez:1
orduña:1
ceo:1
asociación:3
productores:9
empacadores:1
apeam:3
integrada:1
industria:2
aguacatera:1
luego:1
trabajo:1
convertido:2
representante:1
grande:2
siendo:1
destino:2
principal:3
fundó:1
cinco:1
empacadoras:2
huertas:1
exportaron:3
seis:1
cuatro:1
meses:2
hoy:3
fortalece:1
cadena:1
valor:2
producción:3
forma:1
positiva:1
gracias:3
ello:1
mejorado:1
calidad:4
vida:1
integran:1

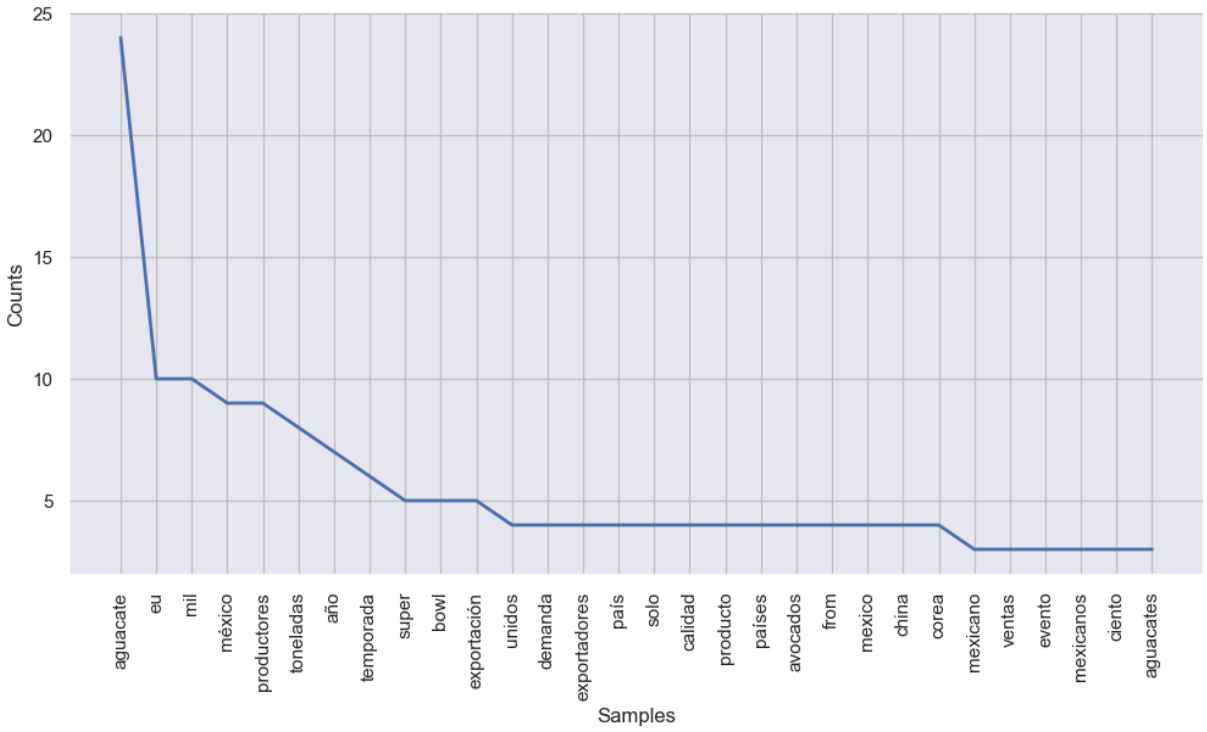
importante:3
además:2
disminuir:1
migración:1
razón:1
producto:4
agropecuario:1
considerado:1
oro:1
verde:1
creció:1
promovido:1
fruto:3
países:4
bajo:1
marca:3
avocados:4
from:4
mexico:4
haciendo:1
ícono:1
internacional:1
campaña:3
aumentó:1
posicionamiento:1
mundo:1
tras:1
colocando:1
principales:2
nivel:2
mundial:2
pasar:1
exportadas:1
principios:1
llegamos:1
seguido:1
canadá:3
japón:3
francia:1
salvador:1
españa:1
honduras:1
holanda:1
china:4
corea:4
sur:3
qatar:1
cuanto:1
consumo:2
estadounidense:3
pasó:1
gramos:1
per:1
cápita:1
kilogramos:1
persona:1

día:1
gran:1
medida:1
esfuerzan:1
constantemente:1
sembrar:1
cosechar:1
limpio:1
importancia:1
nace:1
finalidad:1
llevar:1
número:2
vecino:1
según:2
departamento:1
agricultura:1
cada:2
mercado:1
provienen:1
éxito:1
cosechando:1
haber:1
diseñado:1
estrategias:1
correctas:1
decir:1
dirigir:1
campañas:1
publicitarias:1
específicas:1
público:2
latino:1
volverá:1
aparecer:1
comercial:1
segundos:1
mostrará:1
transmitirá:1
arranque:1
segundo:1
cuarto:1
partido:2
destinamos:1
presupuesto:1
marketing:1
si:1
hacemos:1
posicionan:1
venden:1
cabe:1
destacar:1
dura:1
después:1
viene:1
cuaresma:1

```
mayo:1
fecha:1
incrementa:1
beneficios:1
centran:1
llegan:1
único:1
fresco:1
presente:1
aprobación:1
tan:1
guacamole:2
estrella:1
promedio:2
consume:1
cerca:2
millones:3
cascos:1
futbol:1
deportivo:1
abastecer:1
preparan:1
anticipación:2
incluso:1
apartada:1
vendida:1
considerar:1
buscan:1
alta:1
mediante:1
intervención:1
servicio:1
nacional:1
sanidad:1
inocuidad:1
valida:1
estándares:1
ganarán:1
prevé:1
exportaciones:1
ronde:1
pesos:1
superior:1
casi:2
doble:1
alcanzado:1
pasado:1
perspectiva:1
crecimiento:2
futuro:1
estimaciones:1
esperamos:2
hacia:1
detone:1
producir:1
venta:2
```

```
europa:1
mercados:3
sino:1
intención:1
abrir:1
espacios:1
india:1
chile:1
tocado:1
puertas:1
buena:1
respuesta:1
interés:1
interesa:1
satisfacer:1
crecieron:1
hectáreas:2
plantación:1
cuáles:1
cuentan:1
certificación:1
proceso:1
obtenerlo:1
djr:1
```

In [76]:
```
#Visualizar con limpieza de palabras
sns.set()

# Crear la figura
plt.figure(figsize=(12, 6))
freq_limpia24.plot(30, cumulative=False)

plt.close()
```



In [77]:
```
limpia_palabras25 = palabrasfiltradas25[:]

for palabra in palabrasfiltradas25:

    if palabra in stopwords.words('spanish'):

        limpia_palabras25.remove(palabra)
```

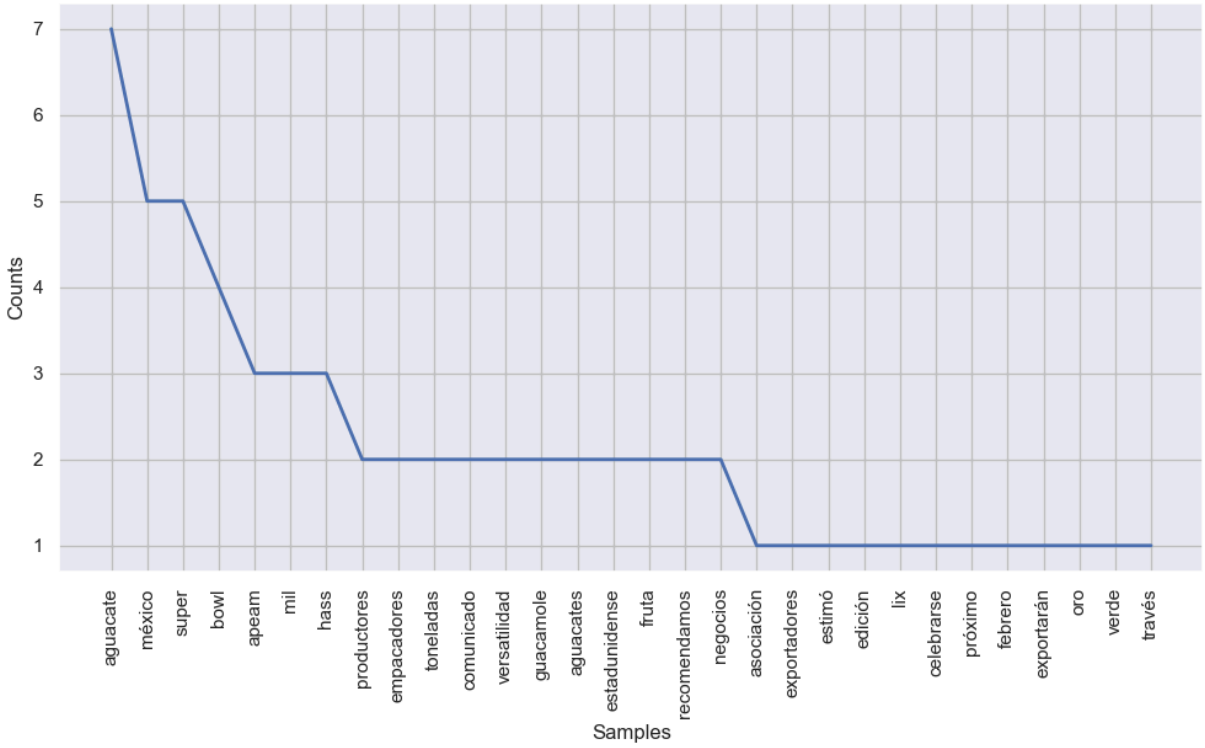
In [78]:
```
#Verificar frecuencia de palabras
freq_limpia25 = nltk.FreqDist(limpia_palabras25)
for key,val in freq_limpia25.items():
    print (str(key) + ':' + str(val))
```

asociación:1
productores:2
empacadores:2
exportadores:1
aguacate:7
méxico:5
apeam:3
estimó:1
edición:1
lix:1
super:5
bowl:4
celebrarse:1
próximo:1
febrero:1
exportarán:1
mil:3
toneladas:2
oro:1
verde:1
través:1
comunicado:2
detalló:1
cifra:1
similar:1
año:1
pasado:1
exportación:1
reflejo:1
arduo:1
trabajo:1
adaptación:1
dedicación:1
miles:1
mexicanos:1
ingenieros:1
agrónomos:1
técnicos:1
campo:1
supervisores:1
seleccionadoras:1
frutas:1
destacó:1
reconocido:1
inigualable:1
sabor:1
textura:1
cremosa:1
versatilidad:2
hass:3
convertido:1
elemento:1
esencial:1
reuniones:1
especialmente:1
protagonista:1

famoso:1
guacamole:2
lee:1
nota:1
informativa:1
precisó:1
mayor:1
productor:1
mundial:1
aguacates:2
evento:1
conlleva:1
impacto:1
economía:1
mexicana:1
estadunidense:2
señaló:1
fruta:2
elige:1
bajo:1
estrictas:1
medidas:1
fitosanidad:1
inocuidad:1
calidad:1
fin:1
llevar:1
mejor:1
espectadores:1
tazón:1
recomendamos:2
canasta:1
básica:1
puebla:1
así:1
incrementado:1
costo:1
años:1
negocios:2
asimismo:1
resaltó:1
unidos:1
principal:1
consumidor:1
siendo:1
ingesta:1
per:1
cápita:1
kilogramos:1
embargo:1
actualmente:1
exportan:1
países:1
mundo:1
favorito:1
acuerdo:1

```
culinaria:1
debido:1
platillos:1
salsas:1
ensaladas:1
mencionó:1
variedad:1
ofrece:1
grasas:1
saludables:1
fibra:1
vitaminas:1
antioxidantes:1
permiten:1
disfrutar:1
culpa:1
finalmente:1
expuso:1
temporada:1
firma:1
avocados:1
from:1
mexico:1
asoció:1
icónico:1
rob:1
gronkowski:1
exjugador:1
profesional:1
fútbol:1
americano:1
apoyar:1
compradores:1
prepararse:1
fiestas:1
aws:1
probará:1
nuevas:1
tecnologías:1
sumarán:1
mdd:1
pib:1
kl:1
```

In [79]:
```
#Visualizar con limpieza de palabras
sns.set()

# Crear la figura
plt.figure(figsize=(12, 6))
freq_limpia25.plot(30, cumulative=False)

plt.close()
```



Análisis Estadístico

In [80]:
```
num_palabras_24 = len(limpia_palabras24)
num_palabras_24_unicas = len(set(limpia_palabras24))
longitud_24_promedio = sum(len(word) for word in limpia_palabras24) / num_palabras_

print(f"Total de palabras Nota del 2024: {num_palabras_24}")
print(f"Palabras únicas Nota del 2024: {num_palabras_24_unicas}")
print(f"Longitud promedio de palabras Nota del 2024: {longitud_24_promedio:.2f}")
```

```
Total de palabras Nota del 2024: 515
Palabras únicas Nota del 2024: 304
Longitud promedio de palabras Nota del 2024: 7.04
```

In [81]:
```
num_palabras_25 = len(limpia_palabras25)
num_palabras_25_unicas = len(set(limpia_palabras25))
longitud_25_promedio = sum(len(word) for word in limpia_palabras25) / num_palabras_

print(f"Total de palabras Nota del 2025: {num_palabras_25}")
print(f"Palabras únicas Nota del 2025: {num_palabras_25_unicas}")
print(f"Longitud promedio de palabras Nota del 2025: {longitud_25_promedio:.2f}")
```

```
Total de palabras Nota del 2025: 189
Palabras únicas Nota del 2025: 155
Longitud promedio de palabras Nota del 2025: 7.34
```

Frecuencia de Palabras y Distribución

In [82]:
```
# Contar frecuencias de palabras
frecuencias_25 = Counter(limpia_palabras25)
```

```
# Obtener las 20 palabras más comunes
palabras_25_comunes = frecuencias_25.most_common(20)
palabras_25, conteos = zip(*palabras_25_comunes)

# Mostrar las palabras y sus conteos
print("Palabras más comunes de la nota del 2025:")
for palabra, conteo in palabras_25_comunes:
    print(f"{palabra}: {conteo}")
```

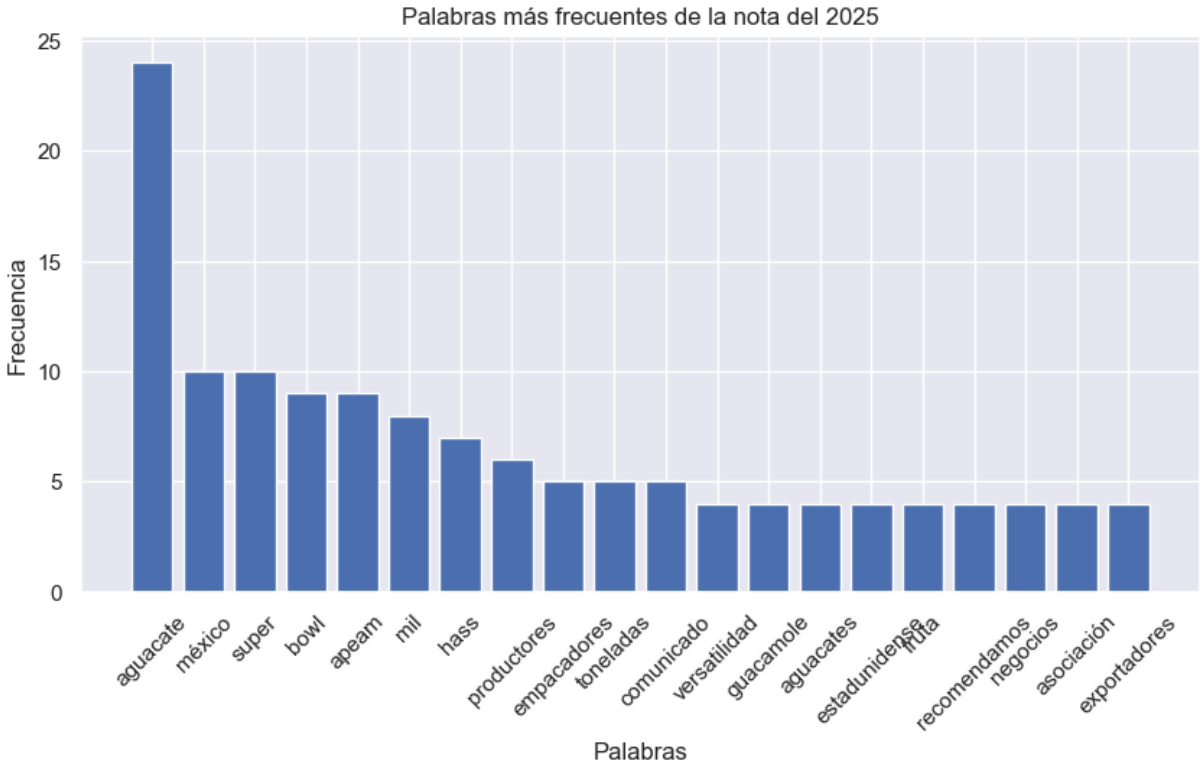
```
Palabras más comunes de la nota del 2025:
aguacate: 7
méxico: 5
super: 5
bowl: 4
apeam: 3
mil: 3
hass: 3
productores: 2
empacadores: 2
toneladas: 2
comunicado: 2
versatilidad: 2
guacamole: 2
aguacates: 2
estadunidense: 2
fruta: 2
recomendamos: 2
negocios: 2
asociación: 1
exportadores: 1
```

In [83]:
```
# Contar frecuencias de palabras
frecuencias_24 = Counter(limpia_palabras24)

# Obtener las 20 palabras más comunes
palabras_24_comunes = frecuencias_24.most_common(20)
palabras_24, conteos = zip(*palabras_24_comunes)

# Mostrar las palabras y sus conteos
print("Palabras más comunes de la nota del 2024:")
for palabra, conteo in palabras_24_comunes:
    print(f"{palabra}: {conteo}")
```

```
Palabras más comunes de la nota del 2024:
aguacate: 24
eu: 10
mil: 10
méxico: 9
productores: 9
toneladas: 8
año: 7
temporada: 6
super: 5
bowl: 5
exportación: 5
unidos: 4
demanda: 4
exportadores: 4
país: 4
solo: 4
calidad: 4
producto: 4
países: 4
avocados: 4
```

In [85]:
```python
# Graficar las palabras más frecuentes
plt.figure(figsize=(10, 5))
plt.bar(palabras_24, conteos)
plt.xticks(rotation=45)
plt.title('Palabras más frecuentes de la nota del 2024')
plt.xlabel('Palabras')
plt.ylabel('Frecuencia')

# Guardar la gráfica en un archivo
plt.savefig(f'C:/Users/PC/Documents/Procesamiento y Clasificacion de  Datos/PC_Dato
# Mostrar la gráfica
plt.show()
plt.close()
```

In [86]:
```
# Graficar las palabras más frecuentes
plt.figure(figsize=(10, 5))
plt.bar(palabras_25, conteos)
plt.xticks(rotation=45)
plt.title('Palabras más frecuentes de la nota del 2025')
plt.xlabel('Palabras')
plt.ylabel('Frecuencia')
# Guardar la gráfica en un archivo
plt.savefig(f'C:/Users/PC/Documents/Procesamiento y Clasificacion de  Datos/PC_Dato
# Mostrar la gráfica
plt.show()
plt.close()
```

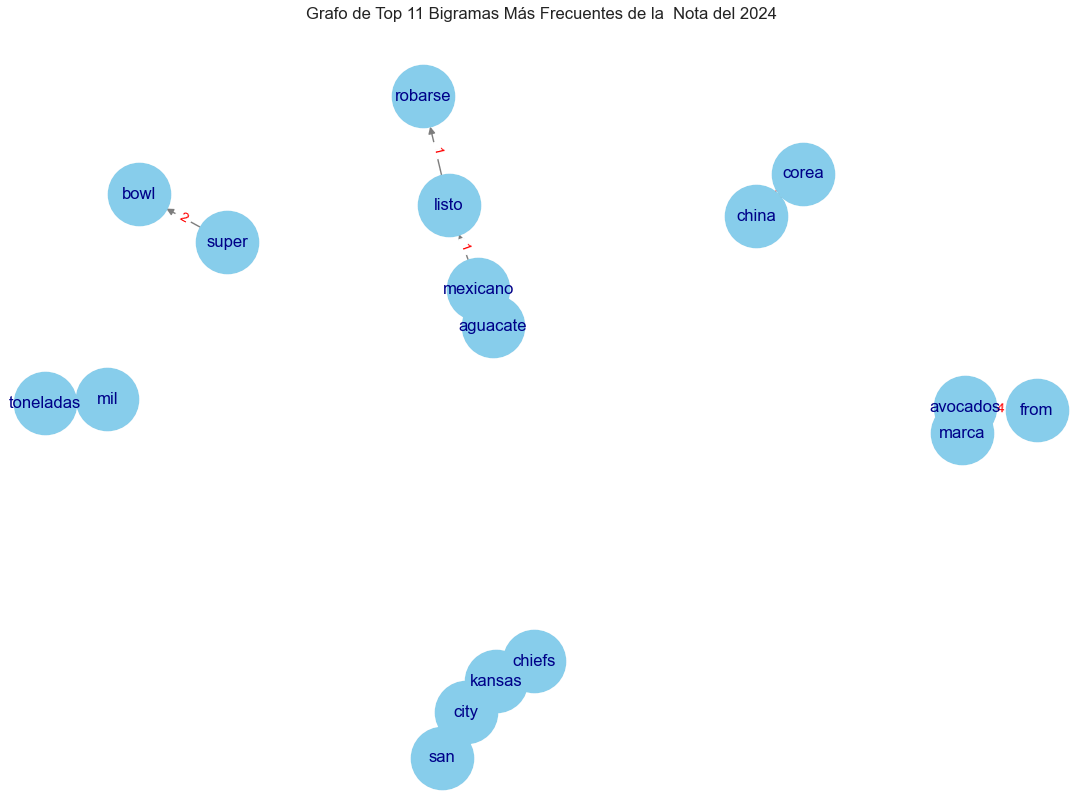


Análisis de N-Gramas

In [87]:
```
# Generar bigramas y trigramas
bigrams_24 = list(ngrams(limpia_palabras24, 3))

# Contar frecuencias de bigramas y trigramas
frecuencia_bigrams_24 = Counter(bigrams_24)

# Imprimir Bigramas más comunes con saltos de línea
print("Bigramas 24 más comunes:")
for bigrams_24, frecuencia in frecuencia_bigrams_24.most_common(15):
    print(f"{bigrams_24}: {frecuencia}")
```

```
Bigramas 24 más comunes:
('avocados', 'from', 'mexico'): 4
('mil', 'toneladas', 'aguacate'): 3
('super', 'bowl', 'lviii'): 2
('chiefs', 'kansas', 'city'): 2
('kansas', 'city', 'san'): 2
('city', 'san', 'francisco'): 2
('marca', 'avocados', 'from'): 2
('china', 'corea', 'sur'): 2
('aguacate', 'mexicano', 'listo'): 1
('mexicano', 'listo', 'robarse'): 1
('listo', 'robarse', 'show'): 1
('robarse', 'show', 'super'): 1
('show', 'super', 'bowl'): 1
('bowl', 'lviii', 'chiefs'): 1
('lviii', 'chiefs', 'kansas'): 1
```

In [90]:
```
# Crear el grafo dirigido
G = nx.DiGraph()

# Filtrar el top N bigramas más frecuentes
top_n = 11
bigramas_top = frecuencia_bigrams_24.most_common(top_n)

# Agregar solo los bigramas del top N al grafo
for bigrama, peso in bigramas_top:
    G.add_edge(bigrama[0], bigrama[1], weight=peso)

# Configuración del layout
pos = nx.spring_layout(G, seed=42)  # Controlar el layout con un seed para reproduc

# Dibujar el grafo
plt.figure(figsize=(12, 8))
nx.draw(
    G,
    pos,
    with_labels=True,
    node_size=2000,
    node_color="skyblue",
    font_size=12,
    font_color="darkblue",
    edge_color="gray",
)

# Agregar etiquetas de peso a las aristas
edge_labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_color='red')

plt.title(f"Grafo de Top {top_n} Bigramas Más Frecuentes de la  Nota del 2024")

# Guardar la gráfica en un archivo
plt.savefig(f'C:/Users/PC/Documents/Procesamiento y Clasificacion de  Datos/PC_Dato
# Mostrar la gráfica
plt.show()
plt.close()
```

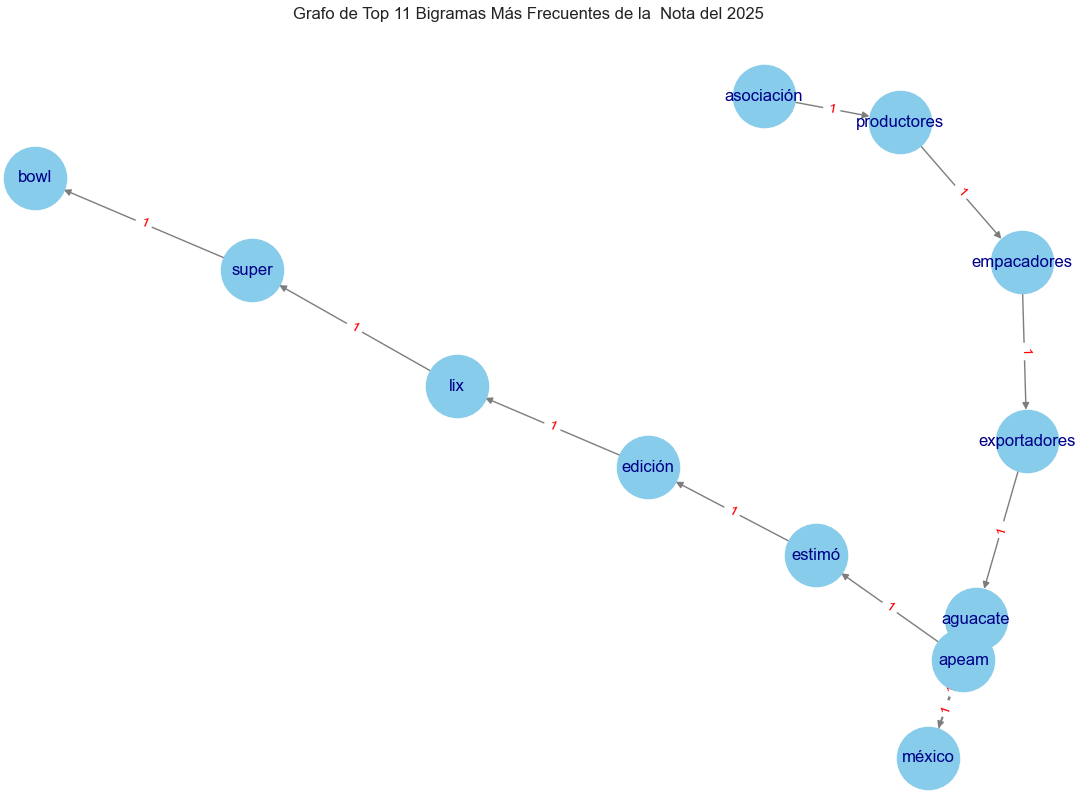


In [91]:
```
# Generar bigramas y trigramas
bigrams_25 = list(ngrams(limpia_palabras25, 3))

# Contar frecuencias de bigramas y trigramas
frecuencia_bigrams_25 = Counter(bigrams_25)

# Imprimir Bigramas más comunes con saltos de línea
print("Bigramas 25 más comunes:")
for bigrams_25, frecuencia in frecuencia_bigrams_25.most_common(15):
    print(f"{bigrams_25}: {frecuencia}")
```

```
Bigramas 25 más comunes:
('asociación', 'productores', 'empacadores'): 1
('productores', 'empacadores', 'exportadores'): 1
('empacadores', 'exportadores', 'aguacate'): 1
('exportadores', 'aguacate', 'méxico'): 1
('aguacate', 'méxico', 'apeam'): 1
('méxico', 'apeam', 'estimó'): 1
('apeam', 'estimó', 'edición'): 1
('estimó', 'edición', 'lix'): 1
('edición', 'lix', 'super'): 1
('lix', 'super', 'bowl'): 1
('super', 'bowl', 'celebrarse'): 1
('bowl', 'celebrarse', 'próximo'): 1
('celebrarse', 'próximo', 'febrero'): 1
('próximo', 'febrero', 'exportarán'): 1
('febrero', 'exportarán', 'mil'): 1
```

In [92]:
```
# Crear el grafo dirigido
G = nx.DiGraph()
```

```
# Filtrar el top N bigramas más frecuentes
top_n = 11
bigramas_top = frecuencia_bigrams_25.most_common(top_n)

# Agregar solo los bigramas del top N al grafo
for bigrama, peso in bigramas_top:
    G.add_edge(bigrama[0], bigrama[1], weight=peso)

# Configuración del layout
pos = nx.spring_layout(G, seed=42)  # Controlar el layout con un seed para reproduc

# Dibujar el grafo
plt.figure(figsize=(12, 8))
nx.draw(
    G,
    pos,
    with_labels=True,
    node_size=2000,
    node_color="skyblue",
    font_size=12,
    font_color="darkblue",
    edge_color="gray",
)

# Agregar etiquetas de peso a las aristas
edge_labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_color='red')

plt.title(f"Grafo de Top {top_n} Bigramas Más Frecuentes de la  Nota del 2025 ")

# Guardar la gráfica en un archivo
plt.savefig(f'C:/Users/PC/Documents/Procesamiento y Clasificacion de  Datos/PC_Dato
# Mostrar la gráfica
plt.show()
plt.close()
```



Análisis de Signos de Puntuación

In [93]:
```
# Encontrar todos los signos de puntuación en el texto original
signos_24 = re.findall(r'[^\w\s]', texto24)

# Contar la frecuencia de cada signo de puntuación
frecuencia_signos_24= Counter(signos_24)

# Mostrar las frecuencias de los signos de puntuación
print("Frecuencia nota del 2024  de signos  de puntuación:")
for signo_24, frecuencia in frecuencia_signos_24.items():
    print(f"'{signo_24}': {frecuencia}")
```

```
Frecuencia nota del 2024  de signos  de puntuación:
',': 70
'(': 2
')': 2
'.': 32
':': 1
'“': 2
'”': 2
'¿': 6
'?': 6
';': 1
```

In [94]:
```
# Encontrar todos los signos de puntuación en el texto original
signos_25 = re.findall(r'[^\w\s]', texto25)

# Contar la frecuencia de cada signo de puntuación
```

```
frecuencia_signos_25= Counter(signos_25)

# Mostrar las frecuencias de los signos de puntuación
print("Frecuencia nota del 2025 de signos  de puntuación:")
for signo_25, frecuencia in frecuencia_signos_25.items():
    print(f"'{signo_25}': {frecuencia}")
```

```
Frecuencia nota del 2025 de signos  de puntuación:
'(': 1
')': 1
',': 31
'.': 16
'“': 2
'”': 2
':': 3
'¿': 1
'?': 1
```

Análisis de emojis

Emojis nota del 2024

In [29]:
```
# Extraer emojis del texto
emojis_encontrados_24 = [char for char in texto24 if emoji.is_emoji(char)]

# Contar la frecuencia de cada emoji
frecuencia_emojis_24 = Counter(emojis_encontrados_24)

# Mostrar los emojis más comunes
print("Emojis más usados nota del 2024:")
for emj, frecuencia in frecuencia_emojis_24.most_common(10):
    print(f"'{emj}': {frecuencia}")
```

```
Emojis más usados nota del 2024:
```

Emojis nota del 2025

In [30]:
```
# Extraer emojis del texto
emojis_encontrados_25 = [char for char in texto25 if emoji.is_emoji(char)]

# Contar la frecuencia de cada emoji
frecuencia_emojis_25 = Counter(emojis_encontrados_25)

# Mostrar los emojis más comunes
print("Emojis más usados nota del 2025:")
for emj, frecuencia in frecuencia_emojis_25.most_common(10):
    print(f"'{emj}': {frecuencia}")
```

```
Emojis más usados nota del 2025:
```

STEMM

Notas del 2024 Raices mas comunes

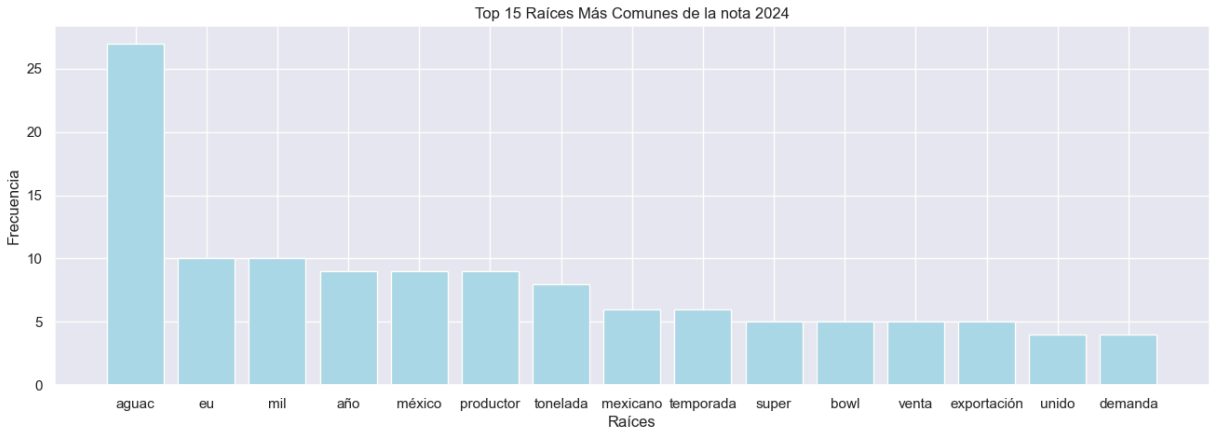
In [95]:
```
# Crear instancia del PorterStemmer
porter = PorterStemmer()

# Aplicar stemming a cada palabra
stems24 = [porter.stem(palabra) for palabra in limpia_palabras24]
```

In [96]:
```
frecuencias24 = Counter(stems24)
```

In [97]:
```
# Mostrar las 10 raíces más comunes
print("Raíces más comunes: \n")
for raiz, frecuencia in frecuencias24.most_common(15):
    print(f"{raiz}: {frecuencia}")
```

```
Raíces más comunes:

aguac: 27
eu: 10
mil: 10
año: 9
méxico: 9
productor: 9
tonelada: 8
mexicano: 6
temporada: 6
super: 5
bowl: 5
venta: 5
exportación: 5
unido: 4
demanda: 4
```

In [98]:
```
# Obtener las 15 raíces más comunes
top_raices = frecuencias24.most_common(15)
raices, conteos = zip(*top_raices)

# Crear gráfico de barras
plt.figure(figsize = (16,5))
plt.bar(raices, conteos, color = 'lightblue')
plt.title("Top 15 Raíces Más Comunes de la nota 2024")
plt.xlabel("Raíces")
plt.ylabel("Frecuencia")
plt.show()

# Guardar la gráfica en un archivo
plt.savefig(f'C:/Users/PC/Documents/Procesamiento y Clasificacion de  Datos/PC_Dato
# Mostrar la gráfica
plt.show()
plt.close()
```



<Figure size 640x480 with 0 Axes>

In [99]:
```
# Crear instancia del PorterStemmer
stemmer = PorterStemmer()

# Aplicar stemming a cada palabra de la lista limpia_palabras24
limpia_palabras_stems24 = [stemmer.stem(token) for token in limpia_palabras24]

# Imprimir el resultado
#print("Palabras originales:", limpia_palabras24)
#print("Palabras con stemming:", limpia_palabras_stems24)
```

In [115…
```
frecuencias24stem = Counter(limpia_palabras_stems24)

# Obtener las 15 raíces más comunes
top_raices = frecuencias24stem.most_common(15)
raices, conteos = zip(*top_raices)

# Crear gráfico de barras
plt.figure(figsize = (16,5))
plt.bar(raices, conteos, color = 'lightblue')
plt.title("Top palabras Más Comunes de la nota 2024 (Con STEMM)")
plt.xlabel("Raíces")
plt.ylabel("Frecuencia")
plt.show()

# Guardar la gráfica en un archivo
plt.savefig(f'C:/Users/PC/Documents/Procesamiento y Clasificacion de  Datos/PC_Dato
# Mostrar la gráfica
plt.show()
plt.close()
```

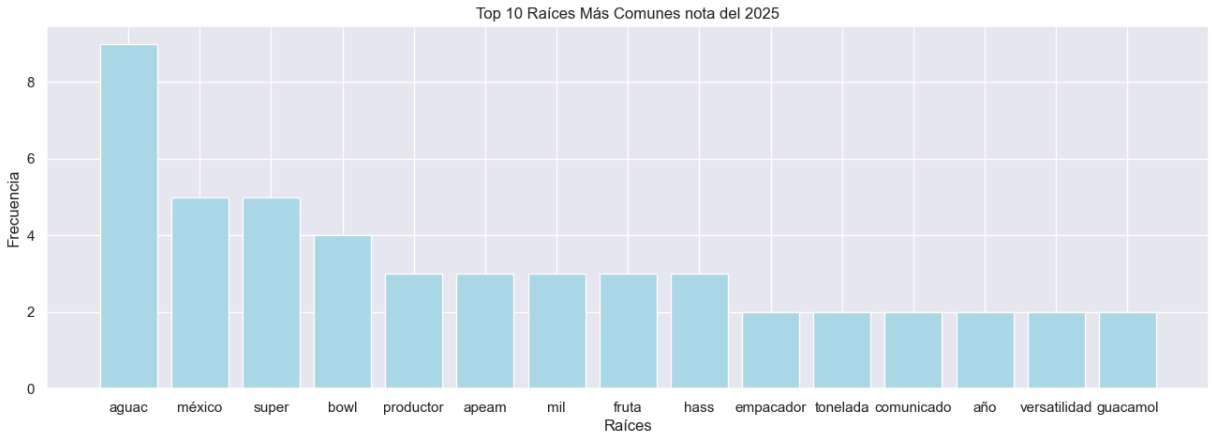

```
<Figure size 640x480 with 0 Axes>
```

Notas del 2025 Raices mas comunes

In [101…
```
# Crear instancia del PorterStemmer
porter = PorterStemmer()

# Aplicar stemming a cada palabra
stems25 = [porter.stem(palabra) for palabra in limpia_palabras25]
```

In [102…
```
frecuencias25 = Counter(stems25)
```

In [103…
```
# Obtener las 10 raíces más comunes
top_raices = frecuencias25.most_common(15)
raices, conteos = zip(*top_raices)

# Crear gráfico de barras
plt.figure(figsize = (16,5))
plt.bar(raices, conteos, color = 'lightblue')
plt.title("Top 10 Raíces Más Comunes nota del 2025")
plt.xlabel("Raíces")
plt.ylabel("Frecuencia")
plt.show()

# Guardar la gráfica en un archivo
plt.savefig(f'C:/Users/PC/Documents/Procesamiento y Clasificacion de  Datos/PC_Dato
# Mostrar la gráfica
plt.show()
plt.close()
```



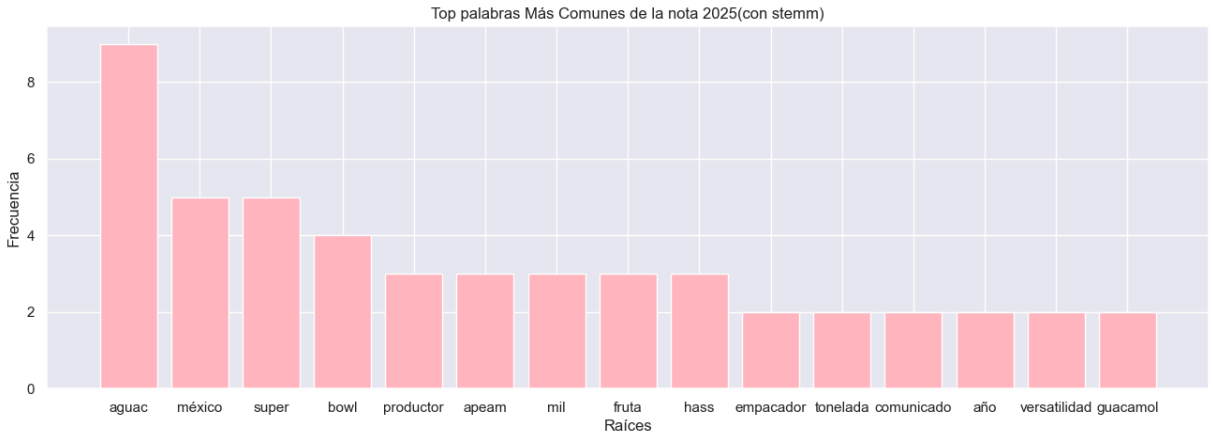

```
<Figure size 640x480 with 0 Axes>
```

In [104…

```python
from nltk.stem import PorterStemmer

# Crear instancia del PorterStemmer
stemmer = PorterStemmer()

# Aplicar stemming a cada palabra de la lista limpia_palabras24
limpia_palabras_stems25 = [stemmer.stem(token) for token in limpia_palabras25]

# Imprimir el resultado
#print("Palabras originales:", limpia_palabras25)
#print("Palabras con stemming:", limpia_palabras_stems25)
```

In [112…

```python
frecuencias25stem = Counter(limpia_palabras_stems25)

# Obtener las 15 raíces más comunes
top_raices = frecuencias25stem.most_common(15)
raices, conteos = zip(*top_raices)

# Crear gráfico de barras
plt.figure(figsize = (16,5))
plt.bar(raices, conteos, color = 'lightpink')
plt.title("Top palabras Más Comunes de la nota 2025(con stemm)")
plt.xlabel("Raíces")
plt.ylabel("Frecuencia")
plt.show()

# Guardar la gráfica en un archivo
plt.savefig(f'C:/Users/PC/Documents/Procesamiento y Clasificacion de  Datos/PC_Dato
# Mostrar la gráfica
plt.show()
plt.close()
```



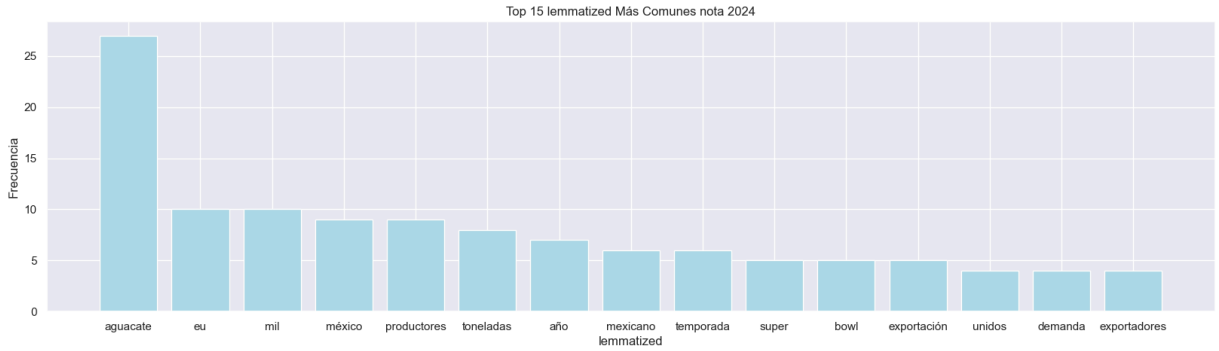

```
<Figure size 640x480 with 0 Axes>
```

Lematizar

In [106…
```python
# Crear instancia del  lemmatizer
lemmatizer = WordNetLemmatizer()

# Aplicar stemming a cada palabra
lemmatized_words = [lemmatizer.lemmatize(palabra) for palabra in limpia_palabras24]
#lemmatized_words
```

In [107…
```python
frecuenciaslemmatized = Counter(lemmatized_words)
```

In [108…
```python
# Obtener las 15 lemmatized más comunes
top_lemmatized = frecuenciaslemmatized.most_common(15)
lemmatized, conteos = zip(*top_lemmatized)

# Crear gráfico de barras
plt.figure(figsize = (20,5))
plt.bar(lemmatized, conteos, color = 'lightblue')
plt.title("Top 15 lemmatized Más Comunes nota 2024")
plt.xlabel("lemmatized")
plt.ylabel("Frecuencia")
plt.show()


# Guardar la gráfica en un archivo
plt.savefig(f'C:/Users/PC/Documents/Procesamiento y Clasificacion de  Datos/PC_Dato
# Mostrar la gráfica
plt.show()
plt.close()
```

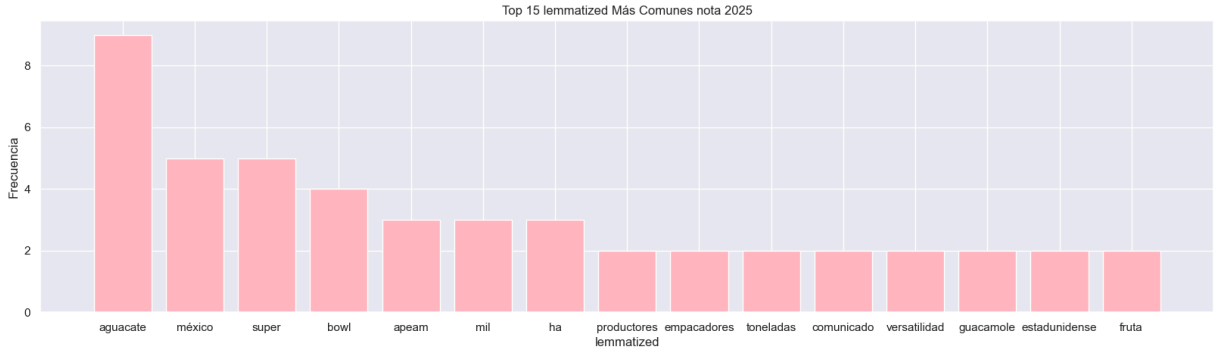



```
<Figure size 640x480 with 0 Axes>
```

In [109…
```python
# Crear instancia del  lemmatizer
lemmatizer = WordNetLemmatizer()

# Aplicar stemming a cada palabra
lemmatized_words = [lemmatizer.lemmatize(palabra) for palabra in limpia_palabras25]
#lemmatized_words
```

In [110…
```python
frecuenciaslemmatized = Counter(lemmatized_words)
```

In [111…
```python
# Obtener las 15 lemmatized más comunes
top_lemmatized = frecuenciaslemmatized.most_common(15)
lemmatized, conteos = zip(*top_lemmatized)

# Crear gráfico de barras
plt.figure(figsize = (20,5))
plt.bar(lemmatized, conteos, color = 'lightpink')
plt.title("Top 15 lemmatized Más Comunes nota 2025")
plt.xlabel("lemmatized")
plt.ylabel("Frecuencia")
plt.show()


# Guardar la gráfica en un archivo
plt.savefig(f'C:/Users/PC/Documents/Procesamiento y Clasificacion de  Datos/PC_Dato
# Mostrar la gráfica
plt.show()
plt.close()
```




```
<Figure size 640x480 with 0 Axes>
```

In [ ]:
```python

```