In [1]:
```
"""Ankita Biswas (ab8ky@virginia.edu)
DS5001
06 May 2022"""
```

Out[1]:
```
'Ankita Biswas (ab8ky@virginia.edu)\nDS5001\n06 May 2022'
```

In [ ]:
```
import pandas as pd
import re
```

Document 1

In [ ]:
```
with open ('1.txt',encoding='utf-8-sig') as f:
    lines = f.read().strip().splitlines()
```

In [ ]:
```
lines
```

In [ ]:
```
len(lines)
```

Remove beginning lines:

idx1 =[] for i in range(len(lines)): if lines[i] == 'Chemical interaction': idx1.append(i)

lines = lines[idx1[0]+1:]

In [ ]:
```
for i in lines:
    i = i.lower()
    i = i.encode("ascii", "ignore").decode()
    i = re.sub('https*\S+', '', i)
    i = re.sub('#\S+', '', i)
    i = re.sub('\'\w+', '', i)
    i = re.sub('\s{2,}', ' ', i)
```

In [ ]:
```
for i in lines:
    print(i)
```

In [ ]:
```
lines
```

In [ ]:
```
len(lines)
```

Remove Special Characters and others

In [ ]:
```
spcChars = ['corresponding author', 'e-mail', 'https', 'fig', 'table']
```

In [ ]:
```
try:
    for i in range(len(lines)):
        for j in spcChars:
            if j in lines[i].lower():
                    del lines[i]
```

```
except:
    'IndexError'
```

In [ ]:
```
len(lines)
```

In [ ]:
```
try:
    for i in range(len(lines)):
        while u'\u00a9' in lines[i]:
            del lines[i]
except:
    'IndexError'
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

def remove_blank_lines(s): lines = s.split("\n") lines = filter(None, lines) return "\n".join(lines)

Remove unnecessary paragraphs (Acknowledgements/References):

In [ ]:
```
idx2=[]
for j in range(len(lines)):
    if 'acknowledgement' in lines[j].lower():
        idx2.append(j)
idx2
```

In [ ]:
```
lines = lines[:idx2[0]]
```

In [ ]:
```
lines
```

In [ ]:
```
# treat ending words
lines_paragraph = []
idx = 0
paragraph = ''
firstSentence = True

while True:
    if idx == len(lines) - 1:
        lines_paragraph.append( paragraph.strip() )
        print('end of text')
        break
    print( idx, lines )

    line = lines[idx].strip()
    end_paragraph = True if len(line) == 0 else False
    if not end_paragraph:
        curr_end = False if line[-1] == '-' else True
    else:
        lines_paragraph.append( paragraph.strip() )
        paragraph = ''
        idx += 1
```

```
        firstSentence = True
        continue

    if idx == 0:
        paragraph += ' ' + line if curr_end else ' ' + line[:-1]
    else:
        prev_line = lines[idx-1].strip()
    if not firstSentence:
        prev_end = False if prev_line[-1] == '-' else True
    else:
        prev_end = True
        firstSentence = False

    if not prev_end and not curr_end:
        paragraph += line[:-1]
    elif not prev_end and curr_end:
        paragraph += line
    elif prev_end and curr_end:
        paragraph += ' ' + line
    elif prev_end and not curr_end:
        paragraph += ' ' + line[:-1]
    idx += 1
```

In [ ]:
```
lines_paragraph
```

In [ ]:
```
while '' in lines_paragraph:
    for i in range(len(lines_paragraph)):
        try:
            if lines_paragraph[i].strip() == '':
                del lines_paragraph[i]
        except:
            'IndexError'
```

In [ ]:
```
lines_paragraph
```

Remove any paragraphs that does not end with '.': Removes equations and other unnecessary lines as well.

In [ ]:
```
try:
    for i in range(len(lines_paragraph)):
        while '.' not in lines_paragraph[i][-1]:
            del lines_paragraph[i]
except:
    'IndexError'
```

In [ ]:
```
len(lines_paragraph)
```

In [ ]:
```
lines_paragraph
```

In [ ]:
```
# remove reference, equations,
ref_regex = '\[([0-9]+|([0-9]+|\,)+)\]'
equt_regex = '\([0-9]+\)'
nonEnglish_regex = '^[^A-Za-z][^A-Za-z0-9/()\-]*'
multispace_regex = ' {2,}'
```

```
lines_paragraph_removed = []
for para_idx, paragraph in enumerate(lines_paragraph):
    sentences = paragraph.split('. ')
    newParagraph = ''
    print( 'paragraph', para_idx )
    paragraph = [x for x in sentences if len(x) != 0]
    for sent_idx, sentence in enumerate(sentences):
    # if len(paragraph) == 1:
    #   if len(paragraph[0]) > 20:
    #     print( 'remove equation', paragraph)
    #     break

        if sent_idx == len(sentences) - 1:
            if sentence[-1] == '.':
                sentence = sentence[:-1]
        sentence = sentence.strip()
        newSentence = re.sub( ref_regex, '', sentence ).strip()
        newSentence = re.sub( equt_regex, '', newSentence ).strip()
        newSentence = re.sub( nonEnglish_regex, '', newSentence ).strip
        newSentence = re.sub( multispace_regex, ' ', newSentence ).stri

        newParagraph += ' ' + newSentence + '.'

    # check newSentence is empty
    print( '', 'sentence', sent_idx )
    print( '', '', sentence)
    print( '', '', newSentence)
    lines_paragraph_removed.append(newParagraph.strip())
```

In [ ]:
```
paras = lines_paragraph_removed
```

In [ ]:
```
len(paras)
```

In [ ]:
```
paras
```

In [ ]:
```
try:
    for i in range(len(paras)):
        count = paras[i].count('.')
        if count < 2:
            del paras[i]
except:
    'IndexError'
```

idx3=[] try: for i in range(len(paras)): count = paras[i].count('.') if count < 4: idx3.append(i) except: 'IndexError'

try: for i in idx3: del paras[i] except: 'IndexError'

In [ ]:
```
len(paras)
```

In [ ]:
```
paras
```

In [ ]:
```
paras_new = []
for i in range(len(paras)):
```

```
    a = paras[i].split('.')
    if a[0][0].isupper() == False:
        a = a[1:]
    b = '.'.join(a)
    paras_new.append(b)
```

In [ ]:
```
while '' in paras_new:
    for i in range(len(paras_new)):
        try:
            if paras_new[i].strip() == '':
                del paras_new[i]
        except:
            'IndexError'
```

In [ ]:
```
paras_new
```

In [ ]:
```
len(paras_new)
```

In [ ]:
```
idx3=[]
try:
    for i in range(len(paras_new)):
        count = paras_new[i].count('.')
        if count < 3:
            idx3.append(i)
except:
    'IndexError'
```

len(idx3)

In [ ]:
```
try:
    for i in idx3:
        del paras_new[i]
except:
    'IndexError'
```

In [ ]:
```
len(paras_new)
```

In [ ]:
```
paras_new
```

In [ ]:
```
OHCO = ['para_num', 'sent_num', 'token_num']
```

In [ ]:
```
PARAS = pd.DataFrame(paras_new, columns=['para_str'])
```

In [ ]:
```
PARAS
```

In [ ]:
```
PARAS.index.names=OHCO[:1]
```

In [ ]:
```
PARAS
```

In [ ]:
```
SENTS = PARAS['para_str'].str.split(r'[.?!;:"]+', expand=True).stack()\
    .to_frame().rename(columns={0:'sent_str'})
SENTS.index.names = OHCO[:2]
SENTS.sent_str = SENTS.sent_str.str.strip()
```

In [ ]:
```
SENTS
```

In [ ]:
```
TOKENS = SENTS['sent_str'].str.split(r"[\s',-]+", expand=True).stack()\
    .to_frame('token_str')
TOKENS.index.names = OHCO[:3]
```

In [ ]:
```
TOKENS
```

In [ ]:
```
TOKENS.to_csv('doc1.csv')
```

Document 2

In [ ]:
```
import pandas as pd
import re
```

In [ ]:
```
with open ('2.txt',encoding='utf-8-sig') as f:
    lines = f.read().strip().splitlines()
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

In [ ]:
```
for i in lines:
    i = i.lower()
    i = i.encode("ascii", "ignore").decode()
    i = re.sub('https*\S+', '', i)
    i = re.sub('#\S+', '', i)
    i = re.sub('\'\w+', '', i)
    i = re.sub('\s{2,}', ' ', i)
```

In [ ]:
```
len(lines)
```

In [ ]:
```
spcChars = ['corresponding author', 'e-mail', 'https', 'fig', 'table']
```

In [ ]:
```
try:
    for i in range(len(lines)):
        for j in spcChars:
            if j in lines[i].lower():
                    del lines[i]

except:
    'IndexError'
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

In [ ]:
```
try:
    for i in range(len(lines)):
        while u'\u00a9' in lines[i]:
            del lines[i]
except:
```

```
    'IndexError'
```

In [ ]:
```
len(lines)
```

In [ ]:
```
idx=[]
for j in range(len(lines)):
    if 'acknowledgement' in lines[j].lower():
        idx.append(j)
idx
```

In [ ]:
```
lines = lines[:idx[0]]
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

In [ ]:
```
# treat ending words
lines_paragraph = []
idx = 0
paragraph = ''
firstSentence = True

while True:
    if idx == len(lines) - 1:
        lines_paragraph.append( paragraph.strip() )
        print('end of text')
        break
    print( idx, lines )

    line = lines[idx].strip()
    end_paragraph = True if len(line) == 0 else False
    if not end_paragraph:
        curr_end = False if line[-1] == '-' else True
    else:
        lines_paragraph.append( paragraph.strip() )
        paragraph = ''
        idx += 1
        firstSentence = True
        continue

    if idx == 0:
        paragraph += ' ' + line if curr_end else ' ' + line[:-1]
    else:
        prev_line = lines[idx-1].strip()
    if not firstSentence:
        prev_end = False if prev_line[-1] == '-' else True
    else:
        prev_end = True
        firstSentence = False

    if not prev_end and not curr_end:
        paragraph += line[:-1]
    elif not prev_end and curr_end:
        paragraph += line
    elif prev_end and curr_end:
        paragraph += ' ' + line
```

```
    elif prev_end and not curr_end:
        paragraph += ' ' + line[:-1]
    idx += 1
```

In [ ]:
```
len(lines_paragraph)
```

In [ ]:
```
lines_paragraph
```

In [ ]:
```
while '' in lines_paragraph:
    for i in range(len(lines_paragraph)):
        try:
            if lines_paragraph[i].strip() == '':
                del lines_paragraph[i]
        except:
            'IndexError'
```

In [ ]:
```
len(lines_paragraph)
```

In [ ]:
```
lines_paragraph
```

In [ ]:
```
# remove reference, equations,
ref_regex = '\[([0-9]+|([0-9]+|\,)+)\]'
equt_regex = '\([0-9]+\)'
nonEnglish_regex = '^[^A-Za-z][^A-Za-z0-9/()\-]*'
multispace_regex = ' {2,}'
lines_paragraph_removed = []
for para_idx, paragraph in enumerate(lines_paragraph):
    sentences = paragraph.split('. ')
    newParagraph = ''
    print( 'paragraph', para_idx )
    paragraph = [x for x in sentences if len(x) != 0]
    for sent_idx, sentence in enumerate(sentences):
    # if len(paragraph) == 1:
    #   if len(paragraph[0]) > 20:
    #     print( 'remove equation', paragraph)
    #     break

        if sent_idx == len(sentences) - 1:
            if sentence[-1] == '.':
                sentence = sentence[:-1]
        sentence = sentence.strip()
        newSentence = re.sub( ref_regex, '', sentence ).strip()
        newSentence = re.sub( equt_regex, '', newSentence ).strip()
        newSentence = re.sub( nonEnglish_regex, '', newSentence ).strip
        newSentence = re.sub( multispace_regex, ' ', newSentence ).stri

        newParagraph += ' ' + newSentence + '.'

    # check newSentence is empty
    print( '', 'sentence', sent_idx )
    print( '', '', sentence)
    print( '', '', newSentence)
    lines_paragraph_removed.append(newParagraph.strip())
```

In [ ]:
```
len(lines_paragraph_removed)
```

In [ ]:
```
paras = lines_paragraph_removed
```

In [ ]:
```
idx1 = ['Materials Letters 232 160-162 Materials Letters 232 160-162.',
        'High entropy alloys Mechanical alloying Nanocrystalline materi
idx2 = []
for i in idx1:
    for j in range(len(paras)):
        if i in paras[j]:
            idx2.append(j)
paras = paras[idx2[1]+1:]
```

In [ ]:
```
idx2
```

In [ ]:
```
paras = paras[idx2[1]+1:]
```

In [ ]:
```
paras
```

In [ ]:
```
drop_list = ['. Introduction.', '. Experimental details.', 'D. Oleszak
             '. Results and discussion.',  '220 50 h + DSC.',
 '00 h.',
 'WMOoNbZrV.',
 '0 h.',
 'IN.',
 'Intensity [a.u.].',
 'theta [deg].',
 'increasing processing time and after heating in the DSC up to 700 °C.
 '61.',  '8.',
 '030.',
 '025.',
 '020.',
 '015.',
 '010.',
 'WMoNbZrV.',
 '.',
 '0 h MA 0,7 6.',
 'a 50 h MA + DSC 700°C.',
 '6.',
 '005 0,5 0,000 -}——— 7 Fe oO _ 00 04 02 03 04 05 06 07 08 009 1,0 — =
 'followed by continuous heating in the DSC up to 700 °C.',
 '62 D. Oleszak et al./ Materials Letters 232 160-162.',
 '174 3,172.',
 '171.',
 '170 3,169 |.',
 '168 3,167.',
 'ml 3,166.',
 '165 1.',
 'aA].',
 'after DSC up to 700°C.',
 '164 3,163.',
 '162 0 10 20 30 40 50 60 70 80 90 100 110.',
 't [h].',
 'treatment.', '. Conclusions.']
```

In [ ]:
```
for i in drop_list:
    paras.remove(i)
```

In [ ]:
```
paras_new = paras
```

idx = [] for i in range(len(paras_new)): if 'extrapolation method' in paras_new[i]: idx.append(i) elif 'diffraction lines and gradual disappearance' in paras_new[i]: idx.append(i) elif 'and Mo reveal the highest melting point' in paras_new[i]: idx.append(i)

In [ ]:
```
idx = []
for i in range(len(paras_new)):
    if 'extrapolation method' in paras_new[i]:
        idx.append(i)

paras_new[4] = paras_new[4].replace(paras_new[4], paras_new[4][:-1] + '
paras_new.remove(paras_new[5])
```

In [ ]:
```
idx
```

In [ ]:
```
paras_new[4] = paras_new[4].replace(paras_new[4], paras_new[4][:-1] + '
paras_new.remove(paras_new[5])
```

while i in idx: paras_new[i-1] = paras_new[i-1].replace(paras_new[i-1], paras_new[i-1][:-1] + ' ' + paras_new[i]) paras_new.remove(paras_new[i])

In [ ]:
```
idx = []
for i in range(len(paras_new)):
    if 'diffraction lines and gradual disappearance' in paras_new[i]:
        idx.append(i)
```

In [ ]:
```
idx
```

In [ ]:
```
paras_new[5] = paras_new[5].replace(paras_new[5], paras_new[5][:-1] + '
paras_new.remove(paras_new[6])
```

In [ ]:
```
idx = []
for i in range(len(paras_new)):
    if 'and Mo reveal the highest melting point' in paras_new[i]:
        idx.append(i)
```

In [ ]:
```
idx
```

In [ ]:
```
paras_new[12] = paras_new[12].replace(paras_new[12], paras_new[12][:-1]
paras_new.remove(paras_new[13])
```

In [ ]:
```
paras_new
```

In [ ]:
```
OHCO = ['para_num', 'sent_num', 'token_num']
PARAS = pd.DataFrame(paras_new, columns=['para_str'])
PARAS.index.names=OHCO[:1]
SENTS = PARAS['para_str'].str.split(r'[.?!;:"]+', expand=True).stack()\
    .to_frame().rename(columns={0:'sent_str'})
SENTS.index.names = OHCO[:2]
SENTS.sent_str = SENTS.sent_str.str.strip()
```

```
TOKENS = SENTS['sent_str'].str.split(r"[\s',-]+", expand=True).stack()\
    .to_frame('token_str')
TOKENS.index.names = OHCO[:3]
TOKENS.to_csv('doc2.csv')
```

In [ ]:
```
TOKENS
```

Document 3

In [ ]:
```
import pandas as pd
import re
```

In [ ]:
```
with open ('3.txt',encoding='utf-8-sig') as f:
    lines = f.read().strip().splitlines()
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

In [ ]:
```
for i in lines:
    i = i.lower()
    i = i.encode("ascii", "ignore").decode()
    i = re.sub('https*\S+', '', i)
    i = re.sub('#\S+', '', i)
    i = re.sub('\'\w+', '', i)
    i = re.sub('\s{2,}', ' ', i)
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

In [ ]:
```
spcChars = ['corresponding author', 'e-mail', 'https', 'fig', 'table']
```

In [ ]:
```
try:
    for i in range(len(lines)):
        for j in spcChars:
            if j in lines[i].lower():
                    del lines[i]

except:
    'IndexError'
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

In [ ]:
```
try:
    for i in range(len(lines)):
        while u'\u00a9' in lines[i]:
            del lines[i]
except:
    'IndexError'
```

```
idx=[]
for j in range(len(lines)):
    if 'acknowledgement' in lines[j].lower():
        idx.append(j)
idx
```

```
lines = lines[:idx[0]]
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

In [ ]:
```
# treat ending words
lines_paragraph = []
idx = 0
paragraph = ''
firstSentence = True

while True:
    if idx == len(lines) - 1:
        lines_paragraph.append( paragraph.strip() )
        print('end of text')
        break
    print( idx, lines )

    line = lines[idx].strip()
    end_paragraph = True if len(line) == 0 else False
    if not end_paragraph:
        curr_end = False if line[-1] == '-' else True
    else:
        lines_paragraph.append( paragraph.strip() )
        paragraph = ''
        idx += 1
        firstSentence = True
        continue

    if idx == 0:
        paragraph += ' ' + line if curr_end else ' ' + line[:-1]
    else:
        prev_line = lines[idx-1].strip()
    if not firstSentence:
        prev_end = False if prev_line[-1] == '-' else True
    else:
        prev_end = True
        firstSentence = False

    if not prev_end and not curr_end:
        paragraph += line[:-1]
    elif not prev_end and curr_end:
        paragraph += line
    elif prev_end and curr_end:
        paragraph += ' ' + line
    elif prev_end and not curr_end:
        paragraph += ' ' + line[:-1]
    idx += 1
```

In [ ]:
```
len(lines_paragraph)
```

In [ ]:
```
lines_paragraph
```

In [ ]:
```
while '' in lines_paragraph:
    for i in range(len(lines_paragraph)):
        try:
            if lines_paragraph[i].strip() == '':
                del lines_paragraph[i]
        except:
            'IndexError'
```

In [ ]:
```
len(lines_paragraph)
```

In [ ]:
```
lines_paragraph
```

In [ ]:
```
idx=[]
for j in range(len(lines_paragraph)):
    if '1. Introduction' in lines_paragraph[j]:
                idx.append(j)
#lines_paragraph = lines_paragraph[idx[1]+1:]
```

In [ ]:
```
idx
```

In [ ]:
```
lines_paragraph = lines_paragraph[21:]
```

In [ ]:
```
lines_paragraph
```

In [ ]:
```
# remove reference, equations,
ref_regex = '\[([0-9]+|([0-9]+|\,)+)\]'
equt_regex = '\([0-9]+\)'
nonEnglish_regex = '^[^A-Za-z][^A-Za-z0-9/()\-]*'
multispace_regex = ' {2,}'
lines_paragraph_removed = []
for para_idx, paragraph in enumerate(lines_paragraph):
    sentences = paragraph.split('. ')
    newParagraph = ''
    print( 'paragraph', para_idx )
    paragraph = [x for x in sentences if len(x) != 0]
    for sent_idx, sentence in enumerate(sentences):
    # if len(paragraph) == 1:
    #   if len(paragraph[0]) > 20:
    #     print( 'remove equation', paragraph)
    #     break

        if sent_idx == len(sentences) - 1:
            if sentence[-1] == '.':
                sentence = sentence[:-1]
        sentence = sentence.strip()
        newSentence = re.sub( ref_regex, '', sentence ).strip()
        newSentence = re.sub( equt_regex, '', newSentence ).strip()
        newSentence = re.sub( nonEnglish_regex, '', newSentence ).strip
        newSentence = re.sub( multispace_regex, ' ', newSentence ).stri

        newParagraph += ' ' + newSentence + '.'
```

```
    # check newSentence is empty
    print( '', 'sentence', sent_idx )
    print( '', '', sentence)
    print( '', '', newSentence)
    lines_paragraph_removed.append(newParagraph.strip())
```

In [ ]:
```
lines_paragraph_removed
```

In [ ]:
```
paras = lines_paragraph_removed
```

In [ ]:
```
drop_list=[ 'Received 30 November 2017; Received in revised form 12 Jan
 'S. Huang et al.',
 'Intermetallics 95 80-84.',
 'List of equiatomic medium- and high-entropy alloys crystallizing in t
        'available experimental and theoretical data for system in the
 '00.',
 '00 T., in the bec phase (K).',
 '00.',
 '200.',
 '1.', 'S. Huang et al.',
 '8 —B—Fe-Fe —@—Cr-Mn —@— Fe-Cr —*— Co-Co —h— Fe-Co —@— Co-Ni —w— Fe-Ni
 '2 —¢-— Fe-Mn —@—Ni-Ni —>— Cr-Cr —&— Ni-Mn —@& Cr-Co —F Mn-Mn.',
 'phase (11,,).',
 '3 5 2 = &.',
 'Reduced exchange interaction (mRy).',
 '6 9.',
 'pth coordination shell.',
 'function of pth coordination shell. The inset shows the magnetic mome
 '80 — 5 _ fo} z = ma Magnetic susceptibility (arb. units) = = wm Ss oa
 '100 200 300 400 fee bee.',
 'Temperature (K) Crystal structure.',
 'fee and bec phases, respectively. The inset shows the magnetic suscep
 'Intermetallics 95 80-84.', '. Results and discussion.',  "Prediction
 'Alloy Wigner-Seitz radius (bohr) Curie temperature (K) Alloy Wigner-S
 'S. Huang et al.', 'Intermetallics 95 80-84.', '. Conclusions.']
```

In [ ]:
```
for i in drop_list:
    paras.remove(i)
```

In [ ]:
```
len(paras)
```

In [ ]:
```
paras
```

In [ ]:
```
for i in paras:
    if 'the multicomponent nature of the HEAs' in i:
        idx = paras.index(i)
result = paras[idx[0]].find('the multicomponent')
```

In [ ]:
```
idx
```

In [ ]:
```
result = paras[4].find('the multicomponent')
```

In [ ]:
```
result
```

In [ ]:
```
paras[4] = paras[4][result:]
```

In [ ]:
```
paras[4]
```

In [ ]:
```
paras
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if 'with a slightly decreasing' in paras[i]:
        idx.append(i)

paras[idx[0]-1] = paras[idx[0]-1].replace(paras[idx[0]-1], paras[idx[0]
paras.remove(paras[idx[0]])
```

In [ ]:
```
paras
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if 'volumes of the alloy components.' in paras[i]:
        idx.append(i)

paras[idx[0]-1] = paras[idx[0]-1].replace(paras[idx[0]-1], paras[idx[0]
paras.remove(paras[idx[0]])
```

In [ ]:
```
paras
```

In [ ]:
```
OHCO = ['para_num', 'sent_num', 'token_num']
PARAS = pd.DataFrame(paras, columns=['para_str'])
PARAS.index.names=OHCO[:1]
SENTS = PARAS['para_str'].str.split(r'[.?!;:"]+', expand=True).stack()\
    .to_frame().rename(columns={0:'sent_str'})
SENTS.index.names = OHCO[:2]
SENTS.sent_str = SENTS.sent_str.str.strip()
TOKENS = SENTS['sent_str'].str.split(r"[\s',-]+", expand=True).stack()\
    .to_frame('token_str')
TOKENS.index.names = OHCO[:3]
TOKENS.to_csv('doc3.csv')
```

In [ ]:
```
TOKENS
```

Document 4

In [ ]:
```
import pandas as pd
import re
```

In [ ]:
```
with open ('4.txt',encoding='utf-8-sig') as f:
    lines = f.read().strip().splitlines()
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

In [ ]:
```
spcChars = ['corresponding author', 'e-mail', 'https']
for i in lines:
    i = i.lower()
    i = i.encode("ascii", "ignore").decode()
    i = re.sub('https*\S+', '', i)
    i = re.sub('#\S+', '', i)
    i = re.sub('\'\w+', '', i)
    i = re.sub('\s{2,}', ' ', i)
```

In [ ]:
```
for i in lines:
    i = i.lower()
    i = i.encode("ascii", "ignore").decode()
    i = re.sub('https*\S+', '', i)
    i = re.sub('#\S+', '', i)
    i = re.sub('\'\w+', '', i)
    i = re.sub('\s{2,}', ' ', i)
```

In [ ]:
```
len(lines)
```

In [ ]:
```
try:
    for i in range(len(lines)):
        for j in spcChars:
            if j in lines[i].lower():
                    del lines[i]

except:
    'IndexError'
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

In [ ]:
```
try:
    for i in range(len(lines)):
        while u'\u00a9' in lines[i]:
            del lines[i]
except:
    'IndexError'
```

In [ ]:
```
len(lines)
```

In [ ]:
```
idx2=[]
for j in range(len(lines)):
    if 'acknowledgement' in lines[j].lower():
        idx2.append(j)
idx2
lines = lines[:idx2[0]]
```

In [ ]:
```
lines = lines[:idx2[0]]
```

In [ ]:
```
len(lines)
```

In [ ]:
```
# treat ending words
lines_paragraph = []
idx = 0
```

```
paragraph = ''
firstSentence = True

while True:
    if idx == len(lines) - 1:
        lines_paragraph.append( paragraph.strip() )
        print('end of text')
        break
    print( idx, lines )

    line = lines[idx].strip()
    end_paragraph = True if len(line) == 0 else False
    if not end_paragraph:
        curr_end = False if line[-1] == '-' else True
    else:
        lines_paragraph.append( paragraph.strip() )
        paragraph = ''
        idx += 1
        firstSentence = True
        continue

    if idx == 0:
        paragraph += ' ' + line if curr_end else ' ' + line[:-1]
    else:
        prev_line = lines[idx-1].strip()
    if not firstSentence:
        prev_end = False if prev_line[-1] == '-' else True
    else:
        prev_end = True
        firstSentence = False

    if not prev_end and not curr_end:
        paragraph += line[:-1]
    elif not prev_end and curr_end:
        paragraph += line
    elif prev_end and curr_end:
        paragraph += ' ' + line
    elif prev_end and not curr_end:
        paragraph += ' ' + line[:-1]
    idx += 1
```

In [ ]:
```
len(lines_paragraph)
```

In [ ]:
```
while '' in lines_paragraph:
    for i in range(len(lines_paragraph)):
        try:
            if lines_paragraph[i].strip() == '':
                del lines_paragraph[i]
        except:
            'IndexError'
```

In [ ]:
```
len(lines_paragraph)
```

In [ ]:
```
# remove reference, equations,
ref_regex = '\[([0-9]+|([0-9]+|\,)+)\]'
equt_regex = '\([0-9]+\)'
```

```
nonEnglish_regex = '^[^A-Za-z][^A-Za-z0-9/()\-]*'
multispace_regex = ' {2,}'
lines_paragraph_removed = []
for para_idx, paragraph in enumerate(lines_paragraph):
    sentences = paragraph.split('. ')
    newParagraph = ''
    print( 'paragraph', para_idx )
    paragraph = [x for x in sentences if len(x) != 0]
    for sent_idx, sentence in enumerate(sentences):
    # if len(paragraph) == 1:
    #   if len(paragraph[0]) > 20:
    #     print( 'remove equation', paragraph)
    #     break

        if sent_idx == len(sentences) - 1:
            if sentence[-1] == '.':
                sentence = sentence[:-1]
        sentence = sentence.strip()
        newSentence = re.sub( ref_regex, '', sentence ).strip()
        newSentence = re.sub( equt_regex, '', newSentence ).strip()
        newSentence = re.sub( nonEnglish_regex, '', newSentence ).strip
        newSentence = re.sub( multispace_regex, ' ', newSentence ).stri

        newParagraph += ' ' + newSentence + '.'

    # check newSentence is empty
    print( '', 'sentence', sent_idx )
    print( '', '', sentence)
    print( '', '', newSentence)
    lines_paragraph_removed.append(newParagraph.strip())
```

In [ ]:
```
len(lines_paragraph_removed)
```

In [ ]:
```
paras = lines_paragraph_removed
```

In [ ]:
```
paras
```

In [ ]:
```
idx2 = []
for j in range(len(paras)):
    if 'ABSTRACT.' in paras[j]:
        idx2.append(j)
```

In [ ]:
```
idx2
```

In [ ]:
```
paras = paras[idx2[0]+1:]
```

In [ ]:
```
paras
```

In [ ]:
```
drop_list = ['. Introduction.', '8 V. Shivam et al. Journal of Alloys a
             '. Materials and experimental details.', '. Results.', 'Int
 '0 (degree).',
 'Fig. . XRD patterns of AlCoCrFeNiMn HEA powders with different millin
 'V. Shivam et al. Journal of Alloys and Compounds 757 87–97 89.', 'Tab
 'Milling time (h) 5h 10h 15h 20h 25h 30h.',
 'Crystallite size(nm) 20 18 17 17 16 15 15 15 Lattice strain (%) 0.54
```

```
'5h 40h.',
'Intensity (arb. units).',
'0 (degree).',
'Fig. . XRD patterns of (a) AlCoCr (b) FeMnNi (c) AlCoCrFeNiMn HEA pow
            '0 V. Shivam et al. Journal of Alloys and Compounds 757 87
           'No. of particles.',
'an as > maa in N = N an.',
'3.0.',
'Particle size (4m).',
'Fig. . Particle size distribution in equiatomic AlCoCrFeNiMn high ent
           'Fig. . SEM micrographs of AlCoCrFeNiMn HEA powders with 40
'surfaces can be observed.',
'V. Shivam et al. Journal of Alloys and Compounds 757 87–97 91.',
'Fig. . TEM bright field image (a) of 40h milled AlCoCrFeNiMn HEA powd
         'B\\ o£ i In (4 = –Rr, + C; (Kissinger equation) E. In(@) = -R
            '2 V. Shivam et al. Journal of Alloys and Compounds 757 87
'i 100 2m.',
"emmee AL's ALLL.",
'a.',
'MY.',
'100 nm.',
'Fig. . The STEM-EDS mapping of equiatomic AlCoCrFeNiMn hexanary high
'BCC phase.',
             'V. Shivam et al. Journal of Alloys and Compounds 757 87–
'DSC/nuV/mg).',
'00 600 800 1000 Temperature (°C).',

          'Fig. . DSC thermogram of 40h milled AlCoCrFeNiMn high entrop
'50° C (1223 K) & 1000 °C (1273 K) have been shown in Fig. 3. It is ev
'. Discussion 4.1. HEAs and BMGs.', 'Table 2.',
'Intensity (arb. units).',
'0 60 90 20 (degree) Fig. . XRD patterns of 40h milled AlCoCrFeNiMn hi
'00°C (773 K). No phase transformation is observed up to this temperat
'FCC .BCC *Mn,Co, # LI, phase.',
'Intensity (arb. units).',
'0 40 60 80 100 20 (degree).',
'Fig. . XRD pattern 40 h milled AlCoCrFeNiMn high entropy alloy powder
     'Activation energy calculation by Kissinger and Ozawa model for t
'Heating Rate , K/min Peak Temperature (K).',
'0 814 20 822 40 838.',
'0 840.',
'Activation Energies kJ/mol-K.',
'10 (Kissinger model) 324 (Ozawa model).',
'4 V. Shivam et al. Journal of Alloys and Compounds 757 87–97.',
'BON alin) ae.',
'Fig. 0. Bright field image and corresponding SADP of AlCoCrFeNiMn pow
'the alloy and the presence of the BCC phase is observed after sinteri
'C00) ge.',
'.',
'.',
'.',
'.',
'Fig. 1. a) Bright field image and (b) corresponding SADP of AlCoCrFeN
'arrows) however not extraordinarily grown due to sluggish diffusion k
           'V. Shivam et al. Journal of Alloys and Compounds 757 87–97
'Fig. 2. Optical micrographs of AlCoCrFeNiMn HEA compact after microwa
'porosity (shown with arrows) can be observed.',
```

```
 'BCC # LI, phase.',
 'Intensity (arb.units).',
 '0 40 60 80 20 (degree).',
 'Fig. 3. XRD patterns of AlCoCrFeNiMn HEA alloy compact after microwav
     'Table 3 The values of atomic size difference 5 (%) in the AlCoCr
 'Element Al Co Cr Fe Ni Mn.',
 'Al _ 6.7 6.8 71 6.9 2.9 Co 6.7 a 0.08 0.4 0.2 3.8 Cr 6.8 0.08 = 0.32
     '6 V. Shivam et al. Journal of Alloys and Compounds 757 87–97.',
 'Table 4 The values of chemical enthalpy of mixing( AH", kJ/mol), of a
 'Element Al Co Cr Fe Ni Mn Al _ –19 –10 -11 –22 -19 Co -19 - –4 -1 0 –
 'Fe –11 -1 -1 – –2 0.',
 'Ni –22 0 –7 –2 - -8 Mn -19 –5 2 0 -8 =.',
 'Table 5.',
 'Calculated thermodynamic and physical parameters for AlCoCrFeMnNi hig
 'alloy. K-1 Q AR mix (Kot) ASconf.',
 '11.39 1944 14.89 2.5.',
 'Tm (K).',
       '3. Thermal stability of the phases evolved in the AlCoCrFeMnNi
```

In [ ]:
```
for i in drop_list:
    paras.remove(i)
```

In [ ]:
```
paras
```

In [ ]:
```
for i in paras:
    if 'solutions. Equiatomic CoCrFeNiMn was the first alloy' in i:
        idx = paras.index(i)
```

In [ ]:
```
result = paras[idx].find('It can be concluded from the present work tha
```

In [ ]:
```
result
```

In [ ]:
```
paras[idx] = paras[idx][result:]
```

In [ ]:
```
paras
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if 'transformation starts beforehand' in paras[i]:
        idx.append(i)

paras[idx[0]-1] = paras[idx[0]-1].replace(paras[idx[0]-1], paras[idx[0]
paras.remove(paras[idx[0]])
```

In [ ]:
```
paras
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if '. Conclusions It can be concluded' in paras[i]:
        idx.append(i)
paras[idx[0]] = paras[idx[0]].replace('. Conclusions ', '')
```

In [ ]:
```
idx = []
for i in range(len(paras)):
```

```
    if 'Visible light micrographs of microwave' in paras[i]:
        idx.append(i)
paras[idx[0]] = paras[idx[0]].replace(paras[idx[0]], paras[idx[0]][:-1]
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if 'It can be concluded from the present work that.' in paras[i]:
        idx.append(i)
    elif '. AlCoCrFeNiMn equiatomic alloy leads to the formation of a s
        idx.append(i)
    elif '. The semi-empirical thermodynamic analysis reveals' in paras
        idx.append(i)
    elif '. The high entropy alloy is stable upto ~500 °C (773 K)' in p
        idx.append(i)
    elif '. Consolidated and sintered pellet of the same alloy after' i
        idx.append(i)
```

In [ ]:
```
paras[idx[0]] = paras[idx[0]].replace(paras[idx[0]], paras[idx[0]] + '
```

In [ ]:
```
paras
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if '. AlCoCrFeNiMn equiatomic alloy leads to the formation' in para
        idx.append(i)
paras=paras[:idx[1]]
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if 'Activation energy calculation for the transformation of this al
        idx.append(i)
```

In [ ]:
```
paras[idx[0]] = paras[idx[0]].replace(paras[idx[0]], paras[idx[0]-1][:-
```

In [ ]:
```
paras[idx[0]-1][:-1]
```

In [ ]:
```
paras.remove(paras[12])
```

In [ ]:
```
paras
```

In [ ]:
```
OHCO = ['para_num', 'sent_num', 'token_num']
PARAS = pd.DataFrame(paras, columns=['para_str'])
PARAS.index.names=OHCO[:1]
SENTS = PARAS['para_str'].str.split(r'[.?!;:"]+', expand=True).stack()\
    .to_frame().rename(columns={0:'sent_str'})
SENTS.index.names = OHCO[:2]
SENTS.sent_str = SENTS.sent_str.str.strip()
TOKENS = SENTS['sent_str'].str.split(r"[\s',-]+", expand=True).stack()\
    .to_frame('token_str')
TOKENS.index.names = OHCO[:3]
TOKENS.to_csv('doc4.csv')
```

In [ ]:
```
TOKENS
```

Document 5

In [ ]:
```
import pandas as pd
import re
```

In [ ]:
```
with open ('5.txt',encoding='utf-8-sig') as f:
    lines = f.read().strip().splitlines()
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

In [ ]:
```
spcChars = ['corresponding author', 'e-mail', 'https']
for i in lines:
    i = i.lower()
    i = i.encode("ascii", "ignore").decode()
    i = re.sub('https*\S+', '', i)
    i = re.sub('#\S+', '', i)
    i = re.sub('\'\w+', '', i)
    i = re.sub('\s{2,}', '', i)

try:
    for i in range(len(lines)):
        for j in spcChars:
            if j in lines[i].lower():
                    del lines[i]

except:
    'IndexError'

for j in [u'\u00a9', u'\u00Ae']:
    try:
        for i in range(len(lines)):
            while  in lines[i]:
                del lines[i]
    except:
        'IndexError'
```

In [ ]:
```

```

In [ ]:
```
try:
    for i in range(len(lines)):
        for j in spcChars:
            if j in lines[i].lower():
                    del lines[i]

except:
    'IndexError'
```

In [ ]:
```
len(lines)
```

In [ ]:
```
try:
    for i in range(len(lines)):
        while u'\u00a9' in lines[i]:
```

```
            del lines[i]
except:
    'IndexError'
```

In [ ]:
```
len(lines)
```

In [ ]:
```
idx2=[]
for j in range(len(lines)):
    if 'acknowledgement' in lines[j].lower():
        idx2.append(j)
idx2
lines = lines[:idx2[0]]
```

In [ ]:
```
len(lines)
```

In [ ]:
```
# treat ending words
lines_paragraph = []
idx = 0
paragraph = ''
firstSentence = True

while True:
    if idx == len(lines) - 1:
        lines_paragraph.append( paragraph.strip() )
        print('end of text')
        break
    print( idx, lines )

    line = lines[idx].strip()
    end_paragraph = True if len(line) == 0 else False
    if not end_paragraph:
        curr_end = False if line[-1] == '-' else True
    else:
        lines_paragraph.append( paragraph.strip() )
        paragraph = ''
        idx += 1
        firstSentence = True
        continue

    if idx == 0:
        paragraph += ' ' + line if curr_end else ' ' + line[:-1]
    else:
        prev_line = lines[idx-1].strip()
    if not firstSentence:
        prev_end = False if prev_line[-1] == '-' else True
    else:
        prev_end = True
        firstSentence = False

    if not prev_end and not curr_end:
        paragraph += line[:-1]
    elif not prev_end and curr_end:
        paragraph += line
    elif prev_end and curr_end:
        paragraph += ' ' + line
    elif prev_end and not curr_end:
```

```
        paragraph += ' ' + line[:-1]
    idx += 1
```

In [ ]:
```
len(lines_paragraph)
```

In [ ]:
```
lines_paragraph
```

In [ ]:
```
while '' in lines_paragraph:
    for i in range(len(lines_paragraph)):
        try:
            if lines_paragraph[i].strip() == '':
                del lines_paragraph[i]
        except:
            'IndexError'
```

In [ ]:
```
len(lines_paragraph)
```

In [ ]:
```
lines_paragraph
```

In [ ]:
```
# remove reference, equations,
ref_regex = '\[([0-9]+|([0-9]+|\,)+)\]'
equt_regex = '\([0-9]+\)'
nonEnglish_regex = '^[^A-Za-z][^A-Za-z0-9/()\-]*'
multispace_regex = ' {2,}'
lines_paragraph_removed = []
for para_idx, paragraph in enumerate(lines_paragraph):
    sentences = paragraph.split('. ')
    newParagraph = ''
    print( 'paragraph', para_idx )
    paragraph = [x for x in sentences if len(x) != 0]
    for sent_idx, sentence in enumerate(sentences):
    # if len(paragraph) == 1:
    #   if len(paragraph[0]) > 20:
    #     print( 'remove equation', paragraph)
    #     break

        if sent_idx == len(sentences) - 1:
            if sentence[-1] == '.':
                sentence = sentence[:-1]
        sentence = sentence.strip()
        newSentence = re.sub( ref_regex, '', sentence ).strip()
        newSentence = re.sub( equt_regex, '', newSentence ).strip()
        newSentence = re.sub( nonEnglish_regex, '', newSentence ).strip
        newSentence = re.sub( multispace_regex, ' ', newSentence ).stri

        newParagraph += ' ' + newSentence + '.'

    # check newSentence is empty
    print( '', 'sentence', sent_idx )
    print( '', '', sentence)
    print( '', '', newSentence)
    lines_paragraph_removed.append(newParagraph.strip())
```

In [ ]:
```
len(lines_paragraph_removed)
```

In [ ]:
```
paras = lines_paragraph_removed
```

In [ ]:
```
paras
```

In [ ]:
```
drop_list=['np} Computational Materials np} Computational Materials.',
 'ARTICLE OPEN.',
 'www.nature.com/npjcompumats.',
 'First-principles-based prediction of yield strength in the RhIrPdPtNi
 "Binglun Yin' and William A. Curtin'.",
       'INTRODUCTION.',   'Laboratory for Multiscale Mechanics Modelin
 'Fédérale de Lausanne, 1015 Lausanne, Switzerland Correspondence: Bing
 'Received: 17 August 2018 Accepted: 10 January 2019 Published online:
 'Published in partnership with the Shanghai Institute of Ceramics of t
 'np} nature partner.',
 'journals.',
 'Np).',
 '.',
 'B. Yin and W.A. Curtin.',
         'T3520 AE, = 2.5785 a b? P36,.',
 'ee Tyo = 0.04865 Pe P363,.',
 'where 6 is the dislocation Burgers vector, and.',
 '2S~ c,AV2/.',
 'is the well-known 6-parameter that describes the collective effect of
 'P= abyyo/11107, Mirosiaa = (Cir — Cra + Cag) /3.',
    'T3520 AE, = 2.5785 a b? P36,.',
 'ee Tyo = 0.04865 Pe P363,.',
 'where 6 is the dislocation Burgers vector, and.',
 '2S~ c,AV2/.',
 'is the well-known 6-parameter that describes the collective effect of
 'P= abyyo/11107, Mirosiaa = (Cir — Cra + Cag) /3.',  '.',
 'elastic constants y°’° and v*”° as 1 + prre P(Cy1, Ci2, Cag) = ae.',
 "Previous applications of the model used experimentally measured isotr
 'of the Voigt-averaged elastic constants UY = (Cy — Ga + 3Caa)/5,.',
 '9, v SBS ee 2(3B + yu") B= (Gi + 2G2)/3,.',
 'where B is the bulk modulus agrees with the full anisotropic model to
 'The resulting strength versus temperature T and strain-rate € is.',
 '.',
 'kgT | €9\\3 ty(T, é) = Tyo(T) ; _ (ear?) | ; where é) =10*s | is a re
 'npj Computational Materials 14.',  'Misfit volumes.', 'AV, = )~ cm(av
 'm.', 'V = Vo + AVnXs, AN,.',
 'Ntt(1 — cp).', 'Published in partnership with the Shanghai Institute
 'Table 1. Compositions and supercell formulas used here to compute mis
 'X, (at.%).',
 'Formula Not.',
 '6 A¢(ABCDEF),7 108 42 A¢(ABCDEF)23 144 0.0 Ao(ABCDEF),3 108 0.0 Ao(AB
     'Published in partnership with the Shanghai Institute of Ceramics
 'B. Yin and W.A. Curtin.', 'Elastic constants.',  '86 176 176 -4 -2 -3
 '96 182 -4 1 -3 288 176 -1 O -1 297 -5 OO -2 286 0 -2 -1 111-2 -2]? 11
 'sym 113, -2 sym 110 —2.',
 '13 112.', 'npj Computational Materials 14.',
 'Np).',
 'np}.',
 'B. Yin and W.A. Curtin.',
 '4 Vo= 13.840 A’ 13.95 £ a= 3.811 A 44Pt A | Z ¥ Pd —~ 13.9F | 7 wT. P
 'Fig. Atomic volumes of random alloys at compositions surrounding the
```

```
 'Table 2. Lattice constant, misfit volumes, and 6-parameter from direc
 "Density functional theory (DFT) Vegard's law.",
 'dp (A).',
 'AVan (A*) AV, (A*) AVwi (A*) AVpq (A”) AVpr (A*) AVcu (A’) 6 (%).',
 '811 0.253 0.767 −2.581 1.412 1.835 −1.686 3.873.',
 '800 0.259 0.799 −2.841 1.605 1.893 −1.715 4.193.',
 "Also show for comparison are results from the application of Vegard's
       'npj Computational Materials 14.',
 'Table 3. Density functional theory (DFT)-computed material.',
 'quantities at T= 0K entering the solute strengthening model, and the
 'dp (A).',
 '(%).',
 'C11 (GPa) Cy2 (GPa) C44 (GPa).',
 'u’ (GPa) v” 90 H110/111 (GPa) 75.',
 'oy (MPa) 583.',
 'The experimentally reported yield strength is shown in parenthesis.',
 'Yield strength prediction.', 'DISCUSSION.',  'Published in partnershi
 'B. Yin and W.A. Curtin.', 'AV, _ V({Cn + (1 _ Gy Ket ~~ CmXs}) 7 V({C
 'Xs.', 'npj Computational Materials 14.',
 'Np).',
 'Np).',
 '.',
 'B. Yin and W.A. Curtin.', 'METHODS DFT methodology.', 'DATA AVAILABIL
 'The data that support the findings of this study are available from t
```

In [ ]:
```
try:
    for i in drop_list:
        paras.remove(i)
except:
    'ValueError'
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if 'computes the fundamental zero-stress energy' in paras[i]:
        idx.append(i)

paras[idx[0]-1] = paras[idx[0]-1].replace(paras[idx[0]-1], paras[idx[0]
paras.remove(paras[idx[0]])
```

In [ ]:
```
paras
```

In [ ]:
```
drop_list2 = ['€', 'Misfit volumes.', 'AV, = )~ cm(avn,.', 'm.', 'V = V
              'Table 1. Compositions and supercell formulas used here to
              'Formula Not.', '6 A¢(ABCDEF),7 108 42 A¢(ABCDEF)23 144 0.
              ', 100 < Nro¢ < 150 and 0.03 < |x,| < 0.06.', 'Published i
              'Elastic constants.', '86 176 176 -4 -2 -3 284 172 171 -1
 '96 182 -4 1 -3 288 176 -1 O -1 297 -5 OO -2 286 0 -2 -1 111-2 -2]? 11
 'sym 113, -2 sym 110 −2.',
 '13 112.', 'np}.', '4 Vo= 13.840 A’ 13.95 £ a= 3.811 A 44Pt A | Z ¥ Pd
              'Fig. Atomic volumes of random alloys at compositions surr
 'Table 2. Lattice constant, misfit volumes, and 6-parameter from direc
 "Density functional theory (DFT) Vegard's law.",
 'dp (A).',
 'AVan (A*) AV, (A*) AVwi (A*) AVpq (A”) AVpr (A*) AVcu (A’) 6 ().',
 '811 0.253 0.767 −2.581 1.412 1.835 −1.686 3.873.',
 '800 0.259 0.799 −2.841 1.605 1.893 −1.715 4.193.',
```

```
 "Also show for comparison are results from the application of Vegard's
 'quantities at T= 0K entering the solute strengthening model, and the
 'dp (A).',
 '().',
 'C11 (GPa) Cy2 (GPa) C44 (GPa).',
 'u' (GPa) v" 90 H110/111 (GPa) 75.',
 'oy (MPa) 583.',
 'The experimentally reported yield strength is shown in parenthesis.',
 'Yield strength prediction.', 'DISCUSSION.', 'Published in partnership
 'Xs.', 'RESULTS',
 'of the Voigt-averaged elastic constants UY = (Cy — Ga + 3Caa)/5,.',
 'The resulting strength versus temperature T and strain-rate  is.',
 '.', 'Stable stacking fault energy and partial separation',
 'The stable stacking fault energy y5 is computed using the ANNNI model
 'where ES? and E*< are the atomic energy for the fully relaxed hcp and
```

In [ ]:
```
for j in drop_list2:
    try:
        for i in range(len(paras)):
            while j in paras[i]:
                paras[i] = paras[i].replace(j, '')
    except:
        'IndexError'
```

In [ ]:
```
while '' in paras:
    for i in range(len(paras)):
        try:
            if paras[i].strip() == '':
                del paras[i]
        except:
            'IndexError'
```

In [ ]:
```
try:
    for i in range(len(paras)):
        while 'desired misfit volumes AV', in paras[i]:
            paras[i] = paras[i].replace('desired misfit volumes AV,', '
except:
    'IndexError'
```

In [ ]:
```
paras
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if 'stress is related to the shear yield stress' in paras[i]:
        idx.append(i)

paras[idx[0]-1] = paras[idx[0]-1].replace(paras[idx[0]-1], paras[idx[0]
paras.remove(paras[idx[0]])
```

In [ ]:
```
paras
```

In [ ]:
```
OHCO = ['para_num', 'sent_num', 'token_num']
PARAS = pd.DataFrame(paras, columns=['para_str'])
PARAS.index.names=OHCO[:1]
SENTS = PARAS['para_str'].str.split(r'[.?!;:"]+', expand=True).stack()\
    .to_frame().rename(columns={0:'sent_str'})
```

```
SENTS.index.names = OHCO[:2]
SENTS.sent_str = SENTS.sent_str.str.strip()
TOKENS = SENTS['sent_str'].str.split(r"[\s',-]+", expand=True).stack()\
    .to_frame('token_str')
TOKENS.index.names = OHCO[:3]
TOKENS.to_csv('doc4.csv')
```

In [ ]:
```
TOKENS
```

Document 6

In [ ]:
```
import pandas as pd
import re
```

In [ ]:
```
with open ('6.txt',encoding='utf-8-sig') as f:
    lines = f.read().strip().splitlines()
```

In [ ]:
```
len(lines)
```

In [ ]:
```
spcChars = ['corresponding author', 'e-mail', 'https']
for i in lines:
    i = i.lower()
    i = i.encode("ascii", "ignore").decode()
    i = re.sub('https*\S+', '', i)
    i = re.sub('#\S+', '', i)
    i = re.sub('\'\w+', '', i)
    i = re.sub('\s{2,}', '', i)

try:
    for i in range(len(lines)):
        for j in spcChars:
            if j in lines[i].lower():
                    del lines[i]

except:
    'IndexError'

for j in [u'\u00a9', u'\u00Ae']:
    try:
        for i in range(len(lines)):
            while j in lines[i]:
                del lines[i]
    except:
        'IndexError'
```

In [ ]:
```
len(lines)
```

In [ ]:
```
idx2=[]
for j in range(len(lines)):
    if 'acknowledgement' in lines[j].lower():
        idx2.append(j)
idx2
lines = lines[:idx2[0]]
```

In [ ]:
```
len(lines)
```

In [ ]:
```
lines
```

In [ ]:
```
# treat ending words
lines_paragraph = []
idx = 0
paragraph = ''
firstSentence = True

while True:
    if idx == len(lines) - 1:
        lines_paragraph.append( paragraph.strip() )
        print('end of text')
        break
    print( idx, lines )

    line = lines[idx].strip()
    end_paragraph = True if len(line) == 0 else False
    if not end_paragraph:
        curr_end = False if line[-1] == '-' else True
    else:
        lines_paragraph.append( paragraph.strip() )
        paragraph = ''
        idx += 1
        firstSentence = True
        continue

    if idx == 0:
        paragraph += ' ' + line if curr_end else ' ' + line[:-1]
    else:
        prev_line = lines[idx-1].strip()
    if not firstSentence:
        prev_end = False if prev_line[-1] == '-' else True
    else:
        prev_end = True
        firstSentence = False

    if not prev_end and not curr_end:
        paragraph += line[:-1]
    elif not prev_end and curr_end:
        paragraph += line
    elif prev_end and curr_end:
        paragraph += ' ' + line
    elif prev_end and not curr_end:
        paragraph += ' ' + line[:-1]
    idx += 1
```

In [ ]:
```
lines_paragraph
```

In [ ]:
```
while '' in lines_paragraph:
    for i in range(len(lines_paragraph)):
        try:
            if lines_paragraph[i].strip() == '':
                del lines_paragraph[i]
        except:
            'IndexError'
```

In [ ]:
```
lines_paragraph
```

In [ ]:
```
# remove reference, equations,
ref_regex = '\[([0-9]+|([0-9]+|\,)+)\]'
equt_regex = '\([0-9]+\)'
nonEnglish_regex = '^[^A-Za-z][^A-Za-z0-9/()\-]*'
multispace_regex = ' {2,}'
lines_paragraph_removed = []
for para_idx, paragraph in enumerate(lines_paragraph):
    sentences = paragraph.split('. ')
    newParagraph = ''
    print( 'paragraph', para_idx )
    paragraph = [x for x in sentences if len(x) != 0]
    for sent_idx, sentence in enumerate(sentences):
    # if len(paragraph) == 1:
    #   if len(paragraph[0]) > 20:
    #     print( 'remove equation', paragraph)
    #     break

        if sent_idx == len(sentences) - 1:
            if sentence[-1] == '.':
                sentence = sentence[:-1]
        sentence = sentence.strip()
        newSentence = re.sub( ref_regex, '', sentence ).strip()
        newSentence = re.sub( equt_regex, '', newSentence ).strip()
        newSentence = re.sub( nonEnglish_regex, '', newSentence ).strip
        newSentence = re.sub( multispace_regex, ' ', newSentence ).stri

        newParagraph += ' ' + newSentence + '.'

    # check newSentence is empty
    print( '', 'sentence', sent_idx )
    print( '', '', sentence)
    print( '', '', newSentence)
    lines_paragraph_removed.append(newParagraph.strip())
```

In [ ]:
```
paras = lines_paragraph_removed
```

In [ ]:
```
paras
```

In [ ]:
```
drop_list =['Journal of Alloys and Compounds 805 (2019) 1237-1245 Journ
 'ES',
 'Journal of Alloys and Compounds',
 'Contents lists available at ScienceDirect',
 'JOURNAL OF',
 'ALLOYS AND COMPOUNDS',
 'ELSEVIER journal homepage: http://www.elsevier.com/locate/jalcom Tare
 'A.S. Rogachev *":*, $.G. Vadchenko °, N.A. Kochetov °, S. Rouvimov ‘,
 '* Mezhanov Institute of Structural Macrokinetics and Materials Scienc
 '> National University of Science and Technology “MISIS”, Moscow, Russ
 'ARTICLE INFO ABSTRACT',
 'Article history:',
 'Received 20 May 2019 Received in revised form',
 '16 July 2019',
```

```
'Accepted 17 July 2019 Available online 18 July 2019', 'Journal of All
'ES',
'Journal of Alloys and Compounds',
'Contents lists available at ScienceDirect',
'JOURNAL OF',
'ALLOYS AND COMPOUNDS',
'ELSEVIER journal homepage: http://www.elsevier.com/locate/jalcom Tare
'A.S. Rogachev *":*, $.G. Vadchenko °, N.A. Kochetov °, S. Rouvimov ‘,
'* Mezhanov Institute of Structural Macrokinetics and Materials Scienc
'> National University of Science and Technology “MISIS”, Moscow, Russ
'ARTICLE INFO ABSTRACT',
'Article history:',
'Received 20 May 2019 Received in revised form',
'16 July 2019',
'Accepted 17 July 2019 Available online 18 July 2019', '1. Introductio
 '3. Results', 'A.S. Rogachev et al. / Journal of Alloys and Compounds
'| V fcc-Ni solid solution',
'| (200) rs | | ° = Bline J, 90 min HEBM + SPS 2 ° 90 min HEBM__2 2 \\
'\\re (110), Ni (111), Cr (110) Mn (221) I (311), Mn (311) Fe (200), C
'meth nt! Nel reser aL ES',
'30 35 40 45 50 55 60 65 70 75 80 85 90 95 100',
'20, degree',
'Fig. 1. Evolution of XRD patterns from initial mixture to consolidate
 '298 K (after',
'Ss annealing) 2 @ c 1273 K £ = 1073 K 873 K',
'298 K (before',
'annealing',
'42 44. 46 48 50 52 54',
'20, degree',
'Fig. 2. High-temperature XRD patterns of the HEA powder (90min HEBM;
 'TK 9',
'2000 5, ae 4 TM l L 0,362 L 0,8 L0,7 + 0,361 Loe | L + 0,360 LO,5 + r
'qT CT CT : -0,1 4 0,355 0) 50 100 150 200 250 Time, min',
'Fig. 3. (color online) Results of the high-temperature XRD experiment
'1240 A.S. Rogachev et al. / Journal of Alloys and Compounds 805 (2019
'Fig. 4. SEM of the multicomponent powder blend: initial mixture of el
'Fig. 5. TEM image and corresponding electron diffraction patterns: (a
 '4. Discussion', 'A.S. Rogachev et al. / Journal of Alloys and Compou
'Table 1 Interpretation of the Electron Diffraction Patterns presented
'Experimental values',
'After HEBM',
'After HEBM and SPS',
'Ring diameter D, 1/nm d-spacings nm Ring diameter D, 1/nm',
'6.69 7.88 9.62 0.208 9.62 11.04 0.181 11.27 12.45 13.28 15.78 0.127 1
'Fig. 6. HRTEM of the powder sample (after HEBM).',
 'Possible interpretations',
'Hypothetical structures: calculated d-spacings in nm, and',
'(hkl) d-spacingsnm fcc, fcc, bcc, a=0.36nm a=1.05nm a=0.73 nm 0.299 -
'Concentration, at.%',
'0 20 40 60 80 100 120 140 160 Distance along the scanning line, nm',
'180 200',
'jw',
'Concentration, at.%',
'= 1 0 20 40 60 80 100 120 140 160 180 200 Distance along the scanning
'Fig. 7. EDS microanalysis along the scanning lines, that are shown by
'present work. These values are in a reasonable agreement with results
```

```
'1242 A.S. Rogachev et al. / Journal of Alloys and Compounds 805 (2019
'Fig. 8. SEM images of the consolidated sample (HEA + SPS): a - polish
 'Table 2',
'[Lara a SGOT OA a Ms]8LONan)',
'ea D200 a',
'Fig. 9. (color online) HAADF STEM microstructure image of the sample
'Fig. 10. HRTEM of the CoCrFeNiMn matrix phases: fcc phase with a = 0.
'EDS analysis of the compacted material (HEBM + SPS).',
'Polished cross section Concentration, at. %',
'Fe Ni Mn',
'Co Cr Before etching 19.0+0.8 214+04 After etching 18.8+0.4 20.1 + 1.
'23.2+1.2 18.6 + 2.0 17.8+1.7 22.8 +0.3 18.8 + 0.4 19.1+0.4',
'A.S. Rogachev et al. / Journal of Alloys and Compounds 805 (2019) 123
'Fig. 11. HRTEM of the precipitated dispersed phase. Angles: « ~ 70°,
'0.597/ 9 og vO, Oo',
'90 oe',
'3s ie Sef .',
'. oo',
'0.520',
'0.512',
'0.482',
'Fig. 12. (color online). Atomic structure of precipitate: HAADF STEM
'Cc, x, mm/s 6, 10°S J/kg K A 9.6 1000 9,4 92 900 9,0 800 8,8 8,6 700
'300 350 400 450 500 550 #600 650 #700 Temperature, K',
'Fig. 13. Temperature dependence of the specific electrical conductivi
 'A gL: (1)', '1244 A.S. Rogachev et al. / Journal of Alloys and Compo
 'Conclusions', 'Mezhanov Institute of Structural Macrokinetics and Ma
'National University of Science and Technology “MISIS”, Moscow, Russia
'6 July 2019.',
'and formed precipitates. Up to three types of precipitates, with char
'Keywords:.',
'Transition metal alloys Mechanical alloying Atomic scale structure El
'300 nm, have been detected and studied. Electric and thermal properti
'. Introduction.', 'materials Science Russian Academy of Sciences, Che
 'S.G. Vadchenko), kolyan_kochetov@mail.ru (N.A. Kochetov), Sergei.Rou
 '. Materials and methods.', '. Results.',
'The XRD patterns for the initial powder mixture, after 60 and 90 min
'V fcc-Ni solid solution.',
'rs | | ° = Bline J, 90 min HEBM + SPS 2 ° 90 min HEBM__2 2 \\ Cc.',
're , Ni , Cr Mn I , Mn Fe , Cr.',
'0 35 40 45 50 55 60 65 70 75 80 85 90 95 100.',
'0, degree.',
'Fig. . Evolution of XRD patterns from initial mixture to consolidated
'The mechanically alloyed powder was heated up to 1273 K with simultan
'98 K (after.',
'98 K (before.',
'2 44. 6 48 50 52 54.',
'0, degree.',
'Fig. . High-temperature XRD patterns of the HEA powder (90min HEBM; C
'000 5, ae 4 TM l L 0,362 L 0,8 L0,7 + 0,361 Loe | L + 0,360 LO,5 + r
'Fig. . color online) Results of the high-temperature XRD experiments:
'Fig. . SEM of the multicomponent powder blend: initial mixture of ele
'Fig. . TEM image and corresponding electron diffraction patterns: (a)
 '. Discussion.', 'Table 1 Interpretation of the Electron Diffraction
'69 7.88 9.62 0.208 9.62 11.04 0.181 11.27 12.45 13.28 15.78 0.127 15.
'Fig. . HRTEM of the powder sample (after HEBM).',  'hkl) d-spacingsnm
```

```
 '20 40 60 80 100 120 140 160 Distance along the scanning line, nm.',
 '80 200.',
 'Fig. . EDS microanalysis along the scanning lines, that are shown by
  'Fig. . SEM images of the consolidated sample (HEA + SPS): a - polish
'Lara a SGOT OA a Ms]8LONan).',
 'Fig. . color online) HAADF STEM microstructure image of the sample co
 'Fig. 0. HRTEM of the CoCrFeNiMn matrix phases: fcc phase with a = 0.3
 'Polished cross section Concentration, at. .',
 '3.2+1.2 18.6 + 2.0 17.8+1.7 22.8 +0.3 18.8 + 0.4 19.1+0.4.',
 'Fig. 1. HRTEM of the precipitated dispersed phase. Angles: « ~ 70°, B
 '597/ 9 og vO, Oo.',
 '0 oe.',
 's ie Sef.',
 '520.',
 '512.',
 '482.',
 'Fig. 2. color online). Atomic structure of precipitate: HAADF STEM im
 '00 350 400 450 500 550 #600 650 #700 Temperature, K.',
 'Fig. 3. Temperature dependence of the specific electrical conductivit
'A gL: ', '. Conclusions.']
```

In [ ]:
```
for j in drop_list:
    try:
        for i in range(len(paras)):
            while j in paras[i]:
                paras[i] = paras[i].replace(j, '')
    except:
        'IndexError'
```

In [ ]:
```
len(paras)
```

In [ ]:
```
paras
```

In [ ]:
```
while '' in paras:
    for i in range(len(paras)):
        try:
            if paras[i].strip() == '':
                del paras[i]
        except:
            'IndexError'
```

In [ ]:
```
len(paras)
```

In [ ]:
```
paras
```

In [ ]:
```
try:
    for i in range(len(paras)):
        if '. .' in paras[i]:
            del paras[i]
except:
    'IndexError'
```

In [ ]:
```
len(paras)
```

In [ ]:
```
paras
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if 'High-entropy alloy CoCrFeNiMn was produced' in paras[i]:
        idx.append(i)
paras = paras[idx[0]:]
```

In [ ]:
```
for i in range(len(paras)):
    count = len(paras[i])
    if '.' in paras[i] and count ==1:
        del paras[i]
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if 'heat – by electrons plus phonons.' in paras[i]:
        idx.append(i)

paras[idx[0]-1] = paras[idx[0]-1].replace(paras[idx[0]-1], paras[idx[0]
paras.remove(paras[idx[0]])
```

In [ ]:
```
paras
```

In [ ]:
```
idx = []
for i in range(len(paras)):
    if 'be seen that the intensities of diffraction peaks' in paras[i]:
        idx.append(i)

paras[idx[0]] = paras[idx[0]].replace(paras[idx[0]], 'The XRD patterns
```

In [ ]:
```
paras
```

In [ ]:
```
OHCO = ['para_num', 'sent_num', 'token_num']
PARAS = pd.DataFrame(paras, columns=['para_str'])
PARAS.index.names=OHCO[:1]
SENTS = PARAS['para_str'].str.split(r'[.?!;:"]+', expand=True).stack()\
    .to_frame().rename(columns={0:'sent_str'})
SENTS.index.names = OHCO[:2]
SENTS.sent_str = SENTS.sent_str.str.strip()
TOKENS = SENTS['sent_str'].str.split(r"[\s',-]+", expand=True).stack()\
    .to_frame('token_str')
TOKENS.index.names = OHCO[:3]
TOKENS.to_csv('doc6.csv')
```

In [ ]:
```
TOKENS
```

Load Corpus

In [1]:
```
import nltk
import pandas as pd
import numpy as np
```

In [2]:
```
CORPUS = pd.read_csv('corpus.csv', index_col = [0])
```

In [3]:
```
CORPUS
```

Out[3]:

doc_num	parg_num	sent_num	token_num	token_str
0	0	0	0	variation
0	0	0	1	lattice
0	0	0	2	constant
0	0	0	3	alloying
0	0	0	4	elements
...	...	...	...	...
15	28	11	7	four
15	28	11	8	six
15	28	11	9	alloying
15	28	11	10	elements
15	28	11	11	studied

30345 rows Ч 4 columns

In [4]:
```
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/digifort/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [5]:
```
CORPUS2 = pd.read_csv('Ankita_Corpus.csv', index_col= [0])
```

In [6]:
```
CORPUS2
```

Out[6]:

doc_id	para_num	sent_num	token_num	token_str
1	0	0	0	The
1	0	0	1	variation
1	0	0	2	of
1	0	0	3	the
1	0	0	4	lattice
...	...	...	...	...
6	27	2	5	=
6	27	2	6	0
6	27	3	0	356
6	27	3	1	nm)
6	27	4	0	NaN

19814 rows × 4 columns

In [7]:
```
CORPUS2['term_str'] = CORPUS2.token_str.str.replace(r"\W+", "", regex=T
```

In [8]:
```
CORPUS2
```

Out[8]:

doc_id	para_num	sent_num	token_num	token_str	term_str
1	0	0	0	The	the
1	0	0	1	variation	variation
1	0	0	2	of	of
1	0	0	3	the	the
1	0	0	4	lattice	lattice
...	...	...	...	...	...
6	27	2	5	=	
6	27	2	6	0	0
6	27	3	0	356	356
6	27	3	1	nm)	nm
6	27	4	0	NaN	NaN

19814 rows × 5 columns

In [9]:
```
CORPUS2['bool'] = CORPUS2['term_str'].apply(lambda x: 'NaN' if x in sto
```

In [10]:
```
CORPUS2
```

Out[10]:

doc_id	para_num	sent_num	token_num	token_str	term_str	bool
1	0	0	0	The	the	NaN
1	0	0	1	variation	variation	1
1	0	0	2	of	of	NaN
1	0	0	3	the	the	NaN
1	0	0	4	lattice	lattice	1
...	...	...	...	...	...	...
6	27	2	5	=		1
6	27	2	6	0	0	1
6	27	3	0	356	356	1
6	27	3	1	nm)	nm	1
6	27	4	0	NaN	NaN	1

19814 rows × 6 columns

In [11]:
```
CORPUS2 = CORPUS2[CORPUS2['bool'] != 'NaN']
```

In [12]:
```
CORPUS2
```

Out[12]:

doc_id	para_num	sent_num	token_num	token_str	term_str	bool
1	0	0	1	variation	variation	1
1	0	0	4	lattice	lattice	1
1	0	0	5	constant	constant	1
1	0	0	7	alloying	alloying	1
1	0	0	8	elements	elements	1
...	...	...	...	...	...	...
6	27	2	5	=		1
6	27	2	6	0	0	1
6	27	3	0	356	356	1
6	27	3	1	nm)	nm	1
6	27	4	0	NaN	NaN	1

12306 rows × 6 columns

In [13]:
```
CORPUS2 = CORPUS2.dropna(subset=['term_str'])
```

In [14]:
```
CORPUS2
```

Out[14]:

doc_id	para_num	sent_num	token_num	token_str	term_str	bool
1	0	0	1	variation	variation	1
1	0	0	4	lattice	lattice	1
1	0	0	5	constant	constant	1
1	0	0	7	alloying	alloying	1
1	0	0	8	elements	elements	1
...	...	...	...	...	...	...
6	27	2	3	stable	stable	1
6	27	2	5	=		1
6	27	2	6	0	0	1
6	27	3	0	356	356	1
6	27	3	1	nm)	nm	1

12194 rows × 6 columns

In [15]:
```
CORPUS2.drop('bool', axis=1, inplace=True)
```

```
/home/digifort/anaconda3/lib/python3.8/site-packages/pandas/core/fram
e.py:4308: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  return super().drop(
```

In [16]:
```
CORPUS2.drop('token_str', axis=1, inplace=True)
```

In [17]:
```
CORPUS2
```

Out[17]:

doc_id	para_num	sent_num	token_num	term_str
1	0	0	1	variation
1	0	0	4	lattice
1	0	0	5	constant
1	0	0	7	alloying
1	0	0	8	elements
...	...	...	...	...
6	27	2	3	stable
6	27	2	5	
6	27	2	6	0
6	27	3	0	356
6	27	3	1	nm

12194 rows Ч 4 columns

In [18]:
```
CORPUS2.to_csv('Ankita_Corpus_2.csv')
```

Annotate

In [22]:
```
import nltk
```

In [23]:
```
CORPUS_new = CORPUS.token_str\
             .apply(lambda x: pd.Series(nltk.pos_tag(x)))\
             .stack()\
             .to_frame('pos_tuple')
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-23-286345259887> in <module>
----> 1 CORPUS_new = CORPUS.token_str\
      2             .apply(lambda x: pd.Series(nltk.pos_tag(x)))\
      3             .stack()\
      4             .to_frame('pos_tuple')

~/anaconda3/lib/python3.8/site-packages/pandas/core/series.py in apply
(self, func, convert_dtype, args, **kwds)
   4136             else:
   4137                 values = self.astype(object)._values
-> 4138                 mapped = lib.map_infer(values, f, convert=conv
ert_dtype)
   4139
   4140         if len(mapped) and isinstance(mapped[0], Series):

pandas/_libs/lib.pyx in pandas._libs.lib.map_infer()

<ipython-input-23-286345259887> in <lambda>(x)
      1 CORPUS_new = CORPUS.token_str\
----> 2             .apply(lambda x: pd.Series(nltk.pos_tag(x)))\
      3             .stack()\
      4             .to_frame('pos_tuple')

~/anaconda3/lib/python3.8/site-packages/nltk/tag/__init__.py in pos_ta
g(tokens, tagset, lang)
    163     """
    164     tagger = _get_tagger(lang)
--> 165     return _pos_tag(tokens, tagset, tagger, lang)
    166
    167

~/anaconda3/lib/python3.8/site-packages/nltk/tag/__init__.py in _pos_t
ag(tokens, tagset, tagger, lang)
    117     # Throws Error if tokens is of string type
    118     elif isinstance(tokens, str):
--> 119         raise TypeError('tokens: expected a list of strings, g
ot a string')
    120
```

In [19]:
```
VOCAB = CORPUS.token_str.value_counts().to_frame('n').sort_index()
VOCAB.index.name = 'token_str'
VOCAB['n_chars'] = VOCAB.index.str.len()
VOCAB['p'] = VOCAB.n / VOCAB.n.sum()
VOCAB['i'] = -np.log2(VOCAB.p)
```

In [20]:
```
VOCAB
```

Out[20]:

token_str	n	n_chars	p	i
-1	1	2	0.000033	14.889029

token_str	n	n_chars	p	i
-32	2	3	0.000066	13.889029
-a-	2	3	0.000066	13.889029
-m-	1	3	0.000033	14.889029
-mn	21	3	0.000692	10.496711
...	...	...	...	...
zrnbo1omorerulo	1	15	0.000033	14.889029
zrre	1	4	0.000033	14.889029
zrrez	1	5	0.000033	14.889029
ztnbo1o0	1	8	0.000033	14.889029

In [21]:
```
VOCAB['max_pos'] = CORPUS[['token_str','pos']].value_counts().unstack(f
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-21-dd11f13af1b2> in <module>
----> 1 VOCAB['max_pos'] = CORPUS[['token_str','pos']].value_counts().unstack(fill_value=0).idxmax(1)

~/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py in __getitem__(self, key)
   3028             if is_iterator(key):
   3029                 key = list(key)
-> 3030             indexer = self.loc._get_listlike_indexer(key, axis=1, raise_missing=True)[1]
   3031
   3032         # take() does not accept boolean indexers

~/anaconda3/lib/python3.8/site-packages/pandas/core/indexing.py in _get_listlike_indexer(self, key, axis, raise_missing)
   1264             keyarr, indexer, new_indexer = ax._reindex_non_unique(keyarr)
   1265
-> 1266         self._validate_read_indexer(keyarr, indexer, axis, raise_missing=raise_missing)
   1267         return keyarr, indexer
   1268

~/anaconda3/lib/python3.8/site-packages/pandas/core/indexing.py in _validate_read_indexer(self, key, indexer, axis, raise_missing)
   1314             if raise_missing:
   1315                 not_found = list(set(key) - set(ax))
-> 1316                 raise KeyError(f"{not_found} not in index")
   1317
   1318             not_found = key[missing_mask]

KeyError: "['pos'] not in index"
```

In [ ]: