نکته: در کد ارسال شده پرینت هر مرحله و نمایش پلات ها کامنت شده و برای نمایش خروجی نیاز است تا پرینت مربوطه از کامنت خارج شود.

ابتدا فایل تکست موجود را باز میکنیم:

```
with open("autocorrect.txt", "r", encoding="utf-8") as file:
    text = file.read()
    print("Original Text opened")
```

مرحله ۱ پیش پردازش

```
# 1
text = text.lower()
text = re.sub(r'\s+', ' ', text)
tokens = text.split()
text = ' '.join(word for word in tokens if word not in string.punctuation and not word.isdigit)
stop_words = set(stopwords.words('english'))
tokens = [word for word in tokens if word not in stop_words]
tokens = [word for word in tokens if len(word) >= 3]
vocabulary = list(set(tokens))
word_freq = {word: tokens.count(word) for word in vocabulary}
```

جهت انجام پیش پردازش ابتدا تمام متن لور کیس شده، با استفاده از regex بیش از یک فاصله حذف شده.

در مرحله بعدی با یک if توکن هایی که علائم نگارشی و یا اعداد نیستند به عنوان متن استخراج شده و در متغیر txt ذخیره شده اند.

در دو مرحله لیست توکن ها با حذف استاپ وارد ها و کلمات کمتر از ۳ کارکتر اپدیت شده.

دلیل حذف کلمات کمتر از ۳ کارکتر می تواند به دلیل مخفف بودن، احتمال کم، اشتباه املایی باشد.

در مرحله بعد با گرفتن ست توکن ها، دایره لغات ایجاد میشود.

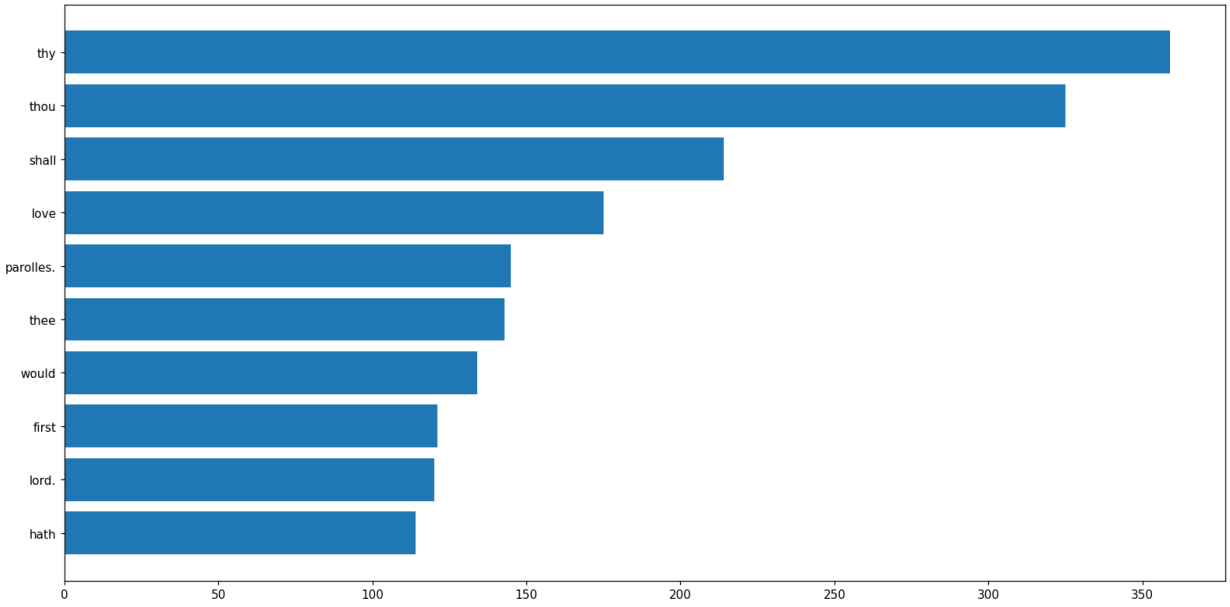
و در دیکشنری word_freq کلید میشود کلمه و تکرار میشود مقدار.

نمایش wordcloud

جهت نمایش ابر کلمات از کد زیر استفاده شده:

```
# wordcloud
preprocessed_text = ' '.join(tokens)
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(preprocessed_text)
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

نمایش نمودار



جهت نمایش نمودار از کد زیر استفاده شد:

```
#most ten words
sorted_word_freq = sorted(word_freq.items(), key=lambda x: x[1], reverse=True)
top_10_words = sorted_word_freq[:10]
words, frequencies = zip(*top_10_words)
plt.figure(figsize=(10, 6))
plt.barh(words, frequencies)
plt.gca().invert_yaxis()
plt.show()
```

مرحله دو محاسبه احتمالات

```
41  # 2
42  def get_count(word_list):
43      word_count = {word: word_list.count(word) for word in word_list}
44      return word_count
45
46  thou_count = get_count(tokens)
47  thou_occurrences = thou_count.get("thou", 0)
48  print(f"thou occurrences is {thou_occurrences}")
49
```

```
PROBLEMS 23  DEBUG CONSOLE  OUTPUT  TERMINAL  COMMENTS

    from nltk.corpus.reader import *
  File "D:\projects\nlp\env_nlp\lib\site-packages\nltk\corpus\reader\__init__.py", line 96, in <module>
    from nltk.corpus.reader.sentiwordnet import *
  File "<frozen importlib._bootstrap>", line 1027, in _find_and_load
  File "<frozen importlib._bootstrap>", line 1006, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 688, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 879, in exec_module
  File "<frozen importlib._bootstrap_external>", line 1012, in get_code
  File "<frozen importlib._bootstrap_external>", line 672, in _compile_bytecode
KeyboardInterrupt
(env_nlp) PS D:\projects\nlp> python nlp.py
Original Text opened
thou occurrences is 325
(env_nlp) PS D:\projects\nlp>
```

همان طور که پیداست تعداد تکرار کلمه thou برابر با ۳۲۵ می باشد.

همچنین احتمال وقوع

```
41  # 2
42  def get_count(word_list):
43      word_count = {word: word_list.count(word) for word in word_list}
44      return word_count
45
46  thou_count = get_count(tokens)
47  thou_occurrences = thou_count.get("thou", 0)
48  # print(f"thou occurrences is {thou_occurrences}")
49
50  def get_probs(word_freq, vocabulary):
51      total_words = sum(word_freq.values())
52      probs = {word: word_freq[word] / total_words for word in vocabulary}
53      return probs
54
55  thou_probs = get_probs(word_freq, vocabulary)
56  thou_probability = thou_probs.get("thou", 0)
57  print(f"thou occurrences is {thou_occurrences} and thou probability is {thou_probability}")
58
```

```
PROBLEMS 23  DEBUG CONSOLE  OUTPUT  TERMINAL  COMMENTS

Original Text opened
thou occurrences is 325
(env_nlp) PS D:\projects\nlp> python nlp.py
Original Text opened
Traceback (most recent call last):
  File "D:\projects\nlp\nlp.py", line 21, in <module>
    word_freq = {word: tokens.count(word) for word in vocabulary}
  File "D:\projects\nlp\nlp.py", line 21, in <dictcomp>
    word_freq = {word: tokens.count(word) for word in vocabulary}
KeyboardInterrupt
(env_nlp) PS D:\projects\nlp> python nlp.py
Original Text opened
thou occurrences is 325 and thou probability is 0.011199944861809912
(env_nlp) PS D:\projects\nlp>
```

همان طور که در تصویر مشاهده میشود امکان وقوع برابر است با ۰.۰۱۱۱۹۹۹۴۴۸۶۱۸۰۹۹۱۲

مرحله ۳ دستکاری رشته ها

تابع delete_char

در پیاده سازی این تابع با استفاده از یک for تمام حالات نبودن یک کارکتر پیاده سازی شده.

و در انتها تمام حالات برای کلمه cans پرینت شده:

```
59  # 3
60  def delete_char(word):
61      deletions = []
62      for i in range(len(word)):
63          deleted_word = word[:i] + word[i + 1:]
64          deletions.append(deleted_word)
65      return deletions
66
67  like_deletions = delete_char("cans")
68  print('delete_char for cans:')
69  print(like_deletions)
```

PROBLEMS 23 DEBUG CONSOLE OUTPUT TERMINAL COMMENTS

```
Traceback (most recent call last):
  File "D:\projects\nlp\nlp.py", line 21, in <module>
    word_freq = {word: tokens.count(word) for word in vocabulary}
  File "D:\projects\nlp\nlp.py", line 21, in <dictcomp>
    word_freq = {word: tokens.count(word) for word in vocabulary}
KeyboardInterrupt
(env_nlp) PS D:\projects\nlp> python nlp.py
Original Text opened
thou occurrences is 325 and thou probability is 0.011199944861809912
(env_nlp) PS D:\projects\nlp> python nlp.py
Original Text opened
delete_char for cans:
['ans', 'cns', 'cas', 'can']
(env_nlp) PS D:\projects\nlp>
```

تابع switch_char

پیاده سازی تابع switch_char مانند تابع قبلی با استفاده از for می باشد:

```python
def switch_char(word):
    switches = []
    for i in range(len(word) - 1):
        switched_word = word[:i] + word[i + 1] + word[i] + word[i + 2:]
        switches.append(switched_word)
    return switches

eta_switches = switch_char("eta")
print('switch_char for eta:')
print(eta_switches)
```

```
PROBLEMS 23  DEBUG CONSOLE  OUTPUT  TERMINAL  COMMENTS

    word_freq = {word: tokens.count(word) for word in vocabulary}
KeyboardInterrupt
(env_nlp) PS D:\projects\nlp> python nlp.py
Original Text opened
thou occurrences is 325 and thou probability is 0.011199944861809912
(env_nlp) PS D:\projects\nlp> python nlp.py
Original Text opened
delete_char for cans:
['ans', 'cns', 'cas', 'can']
(env_nlp) PS D:\projects\nlp> python nlp.py
Original Text opened
switch_char for eta:
['tea', 'eat']
(env_nlp) PS D:\projects\nlp>
```

تابع replace_char

در پیاده سازی این تابع ابتدا حروف مشخص شده اند و با for تمام حالات رخداد به وجود آمده اند:

```python
def replace_char(word):
    alphabet = 'abcdefghijklmnopqrstuvwxyz'
    replacements = []
    for i in range(len(word)):
        for letter in alphabet:
            replaced_word = word[:i] + letter + word[i + 1:]
            replacements.append(replaced_word)
    return replacements

can_replacements = replace_char("can")
print(f'len of can is {len(can_replacements)} and replace_char for can:')
print(can_replacements)
```

```
PROBLEMS 23  DEBUG CONSOLE  OUTPUT  TERMINAL  COMMENTS                    powershell

  File "<frozen importlib._bootstrap_external>", line 1411, in _get_spec
  File "<frozen importlib._bootstrap_external>", line 1577, in find_spec
  File "<frozen importlib._bootstrap_external>", line 161, in _path_isfile
  File "<frozen importlib._bootstrap_external>", line 153, in _path_is_mode_type
  File "<frozen importlib._bootstrap_external>", line 147, in _path_stat
KeyboardInterrupt
(env_nlp) PS D:\projects\nlp> python nlp.py
Original Text opened
len of can is 78 and replace_char for can:
['aan', 'ban', 'can', 'dan', 'ean', 'fan', 'gan', 'han', 'ian', 'jan', 'kan', 'lan', 'man', 'nan', 'oan', 'pan', 'qan', 'ran', 'san', 'tan', 'uan', 'van', 'wan', 'xan', 'yan', 'zan'
, 'can', 'cbn', 'ccn', 'cdn', 'cen', 'cfn', 'cgn', 'chn', 'cin', 'cjn', 'ckn', 'cln', 'cmn', 'cnn', 'con', 'cpn', 'cqn', 'crn', 'csn', 'ctn', 'cun', 'cvn', 'cwn', 'cxn', 'cyn', 'czn
', 'caa', 'cab', 'cac', 'cad', 'cae', 'caf', 'cag', 'cah', 'cai', 'caj', 'cak', 'cal', 'cam', 'can', 'cao', 'cap', 'caq', 'car', 'cas', 'cat', 'cau', 'cav', 'caw', 'cax', 'cay', 'ca
z']
```

که همان طور که در تصویر میبیند تعداد برابر با ۷۸ می باشد.

تابع insert_char

این تابع هم به مشابه تابع قبل پیاده سازی شده:

```python
def insert_char(word):
    alphabet = 'abcdefghijklmnopqrstuvwxyz'
    insertions = []
    for i in range(len(word) + 1):
        for letter in alphabet:
            inserted_word = word[:i] + letter + word[i:]
            insertions.append(inserted_word)
    return insertions

at_insertions = insert_char("at")
print(f'len of at is {len(at_insertions)} and insert_char for at:')
print(at_insertions)
```

```
PROBLEMS 23  DEBUG CONSOLE  OUTPUT  TERMINAL  COMMENTS                    powershell

len of at is 78 and insert_char for at:
['aat', 'bat', 'cat', 'dat', 'eat', 'fat', 'gat', 'hat', 'iat', 'jat', 'kat', 'lat', 'mat', 'nat', 'oat', 'pat', 'qat', 'rat', 'sat', 'tat', 'uat', 'vat', 'wat', 'xat', 'yat', 'zat'
, 'aat', 'abt', 'act', 'adt', 'aet', 'aft', 'agt', 'aht', 'ait', 'ajt', 'akt', 'alt', 'amt', 'ant', 'aot', 'apt', 'aqt', 'art', 'ast', 'att', 'aut', 'avt', 'awt', 'axt', 'ayt', 'azt
', 'ata', 'atb', 'atc', 'atd', 'ate', 'atf', 'atg', 'ath', 'ati', 'atj', 'atk', 'atl', 'atm', 'atn', 'ato', 'atp', 'atq', 'atr', 'ats', 'att', 'atu', 'atv', 'atw', 'atx', 'aty', 'at
z']
(env_nlp) PS D:\projects\nlp> ^C
(env_nlp) PS D:\projects\nlp> python nlp.py
Original Text opened
```

که همان طور که در تصویر میبیند تعداد برابر با ۷۸ می باشد.

تابع edit_one_char

در این تابع توابع پیاده سازی شده فراخوانی شده و در نهایت برای جلوگیری از تکرار ست گرفته شده:

```
# 4
def edit_one_char(word):
    deletes = delete_char(word)
    switches = switch_char(word)
    inserts = insert_char(word)
    replaces = replace_char(word)
    one_edit = list(set(deletes + switches + inserts + replaces))
    return one_edit

at_one_edit = edit_one_char("at")
print(f'len of at is {len(at_one_edit)} edit_one_char for at:')
print(at_one_edit)
```

```
PROBLEMS 23  DEBUG CONSOLE  OUTPUT  TERMINAL  COMMENTS

  File "D:\projects\nlp\nlp.py", line 21, in <module>
    word_freq = {word: tokens.count(word) for word in vocabulary}
  File "D:\projects\nlp\nlp.py", line 21, in <dictcomp>
    word_freq = {word: tokens.count(word) for word in vocabulary}
KeyboardInterrupt
(env_nlp) PS D:\projects\nlp> python nlp.py
Original Text opened
len of at is 130 edit_one_char for at:
['adt', 'atb', 'ato', 'atm', 'gt', 'qat', 'apt', 'ut', 'ag', 'atq', 'aot', 'awt', 'ot', 'atv', 'zt', 'an', 'alt', 'akt', 'aft', 'abt', 'at', 'fat', 'nat', 'kt', 'yt', 'jat', 'ait',
'kat', 'xat', 'dt', 'jt', 'att', 'au', 'ar', 'atp', 'eat', 'ajt', 'cat', 'ast', 'atu', 'av', 'uat', 'az', 'atk', 'ax', 'atg', 'ae', 'a', 'ht', 'ah', 'axt', 'ate', 'lt', 'ai', 'ab',
'ap', 'lat', 'pat', 'tt', 'rt', 'ta', 'aat', 'ati', 'ft', 'sat', 'vt', 'ayt', 'atj', 'wat', 'aj', 'atw', 'amt', 'atl', 'aht', 'af', 'mt', 'rat', 'wt', 'oat', 'ao', 'atn', 'al', 'ak
', 'aet', 'ath', 'aqt', 'aty', 'tat', 'vat', 'dat', 'ant', 'pt', 'st', 'ay', 'hat', 'zat', 'gat', 'mat', 'ata', 'atf', 'agt', 'it', 'xt', 't', 'avt', 'atd', 'bat', 'atc', 'bt', 'ad',
 'ac', 'iat', 'atx', 'atr', 'aa', 'as', 'nt', 'aq', 'yat', 'ct', 'aut', 'et', 'azt', 'ats', 'aw', 'am', 'act', 'qt', 'atz', 'art']
(env_nlp) PS D:\projects\nlp>
```

همان طور که در تصویر میبینید تعداد برابر با ۱۳۰ می باشد.

تابع edit_two_char

ابتدا یک بار edit_one_char فراخوانی شده و برای هر کلمه ایجاد شده مجدد با استفاده از حلقه فراخوانی edit_one_char انجام شده.

```
def edit_two_char(word):
    two_edits = []
    one_edits = edit_one_char(word)
    for one_edit in one_edits:
        two_edits += edit_one_char(one_edit)
    return two_edits

at_two_edit = edit_two_char("at")
print(f'len of at is {len(at_two_edit)} edit_two_char for at:')
print(at_two_edit)
```

```
PROBLEMS 23  DEBUG CONSOLE  OUTPUT  TERMINAL  COMMENTS

Original Text opened
len of at is 20740 edit_two_char for at:
['aer', 'ar', 'aru', 'arj', 'mar', 'far', 'r', 'qr', 'arn', 'arx', 'arr', 'air', 'uar', 'arz', 'car', 'kr', 'ad', 'as', 'lar', 'arf', 'aq', 'rr', 'af', 'vr', 'zar', 'arg', 'mr', 'ar
m', 'tar', 'ae', 'ari', 'kar', 'axr', 'aar', 'aj', 'ap', 'apr', 'arp', 'am', 'ac', 'alr', 'xr', 'ao', 'xar', 'lr', 'anr', 'ur', 'aa', 'cr', 'azr', 'afr', 'yr', 'par', 'aur', 'oar',
'var', 'abr', 'sar', 'au', 'pr', 'awr', 'sr', 'tr', 'az', 'er', 'ay', 'fr', 'yar', 'ary', 'ag', 'ayr', 'rar', 'ark', 'ard', 'har', 'qar', 'ahr', 'arq', 'nar', 'adr', 'ab', 'atr', 'a
vr', 'aro', 'a', 'ara', 'arw', 'ah', 'an', 'ajr', 'hr', 'agr', 'acr', 'arv', 'jr', 'gar', 'arh', 'art', 'aw', 'av', 'akr', 'are', 'arb', 'ir', 'or', 'wr', 'amr', 'nr', 'ak', 'gr', '
dar', 'arl', 'br', 'asr', 'aqr', 'arc', 'iar', 'ear', 'jar', 'war', 'ra', 'bar', 'dr', 'ai', 'aor', 'ars', 'zr', 'at', 'ax', 'al', 'hgt', 'ge', 'kgt', 'gb', 'gv', 'gft', 'gvt', 'gt'
, 'ft', 'gct', 'gtz', 'gtu', 'gc', 'gtq', 'gth', 'gf', 'zgt', 'gkt', 'gz', 'ggt', 'wgt', 'gq', 'pgt', 'gqt', 'gn', 'gpt', 'ngt', 'gmt', 'bgt', 't', 'ygt', 'gtp', 'tt', 'egt', 'nt',
'gtc', 'ut', 'gtj', 'fgt', 'gd', 'qt', 'gtv', 'gyt', 'gtw', 'bt', 'gs', 'xgt', 'rgt', 'igt', 'xt', 'ct', 'wt', 'gx', 'gtd', 'gst', 'gjt', 'gto', 'sgt', 'gtr', 'gwt', 'st', 'gtx', 'g
u', 'rt', 'gnt', 'ugt', 'et', 'gp', 'gi', 'gts', 'mt', 'glt', 'gtb', 'lt', 'gte', 'gta', 'ga', 'jt', 'gh', 'gm', 'gut', 'gtl', 'zt', 'kt', 'dt', 'gk', 'ht', 'tgt', 'gg', 'gdt', 'ot'
, 'gy', 'gty', 'agt', 'gbt', 'gtf', 'gtg', 'mgt', 'ght', 'pt', 'ogt', 'g', 'go', 'gtk', 'gti', 'jgt', 'gzt', 'gr', 'gw', 'gtt', 'it', 'tg', 'cgt', 'vgt', 'lgt', 'gxt', 'gtm', 'yt',
'qgt', 'git', 'got', 'gj', 'get', 'grt', 'at', 'vt', 'gat', 'dgt', 'gl', 'gtn', 'pavt', 'nvt', 'arvt', 'avto', 'avpt', 'uavt', 'avt', 'gvt', 'aht', 'avft', 'avti', 'avn', 'avot', 'a
vtn', 'vat', 'bvt', 'ovt', 'alt', 'savt', 'yavt', 'avtt', 'mvt', 'alvt', 'avst', 'ravt', 'gavt', 'vvt', 'avv', 'apvt', 'avgt', 'davt', 'dvt', 'avet', 'hvt', 'uvt', 'zvt', 'ahvt', 'k
vt', 'avb', 'ava', 'acvt', 'avtc', 'asvt', 'avtm', 'favt', 'avat', 'avh', 'navt', 'avdt', 'avk', 'avu', 'adt', 'aat', 'avtb', 'avtd', 'avtv', 'avut', 'avta', 'avi', 'lavt', 'aot', '
```

همان طور که در تصویر میبینید تعداد برابر با ۲۰۷۴۰ می باشد.

تابع get_seggestions

برای پیاده سازی ابتدا کلمه، احتمال کلمه و لیست کلمات را دریافت کردیم، ابتدا اگر کلمه در لیست کلمات باشد همان کلمه با احتمالش برگردانده میشود. در غیر این صورت edit_one_char و edit_two_char فراخوانی شده و حاصل دو تابع ست شده و تاپلی از کلمات ایجاد شده که کلید کلمه و مقدار تکرار است و اگر تکراری نداشت مقدار ۰ لحاظ می شود. در نهایت به ترتیب نزولی مرتب شده و در یک فایل ذخیره می شود.

```python
# 5
def get_suggestions(word, probs, vocabulary):
    if word in vocabulary:
        return [(word, probs[word])]
    one_edit = edit_one_char(word)
    two_edit = edit_two_char(word)
    candidates = list(set(one_edit + two_edit))
    suggestions = [(candidate, probs.get(candidate, 0)) for candidate in candidates]
    suggestions.sort(key=lambda x: x[1], reverse=True)
    return suggestions

wrte_probs = get_probs(word_freq, vocabulary)
suggestions = get_suggestions("wrte", wrte_probs, vocabulary)

with open("output.txt", "w") as output_file:
    for suggestion in suggestions:
        output_file.write(f"Word: {suggestion[0]}, Probability: {suggestion[1]:.16f}\n")
```

output.txt

```
Word: art, Probability: 0.0018609139155007
Word: true, Probability: 0.0013095320146116
Word: ere, Probability: 0.0007236887449169
Word: worth, Probability: 0.0007236887449169
Word: wife, Probability: 0.0005858432696947
Word: hate, Probability: 0.0004479977944724
Word: write, Probability: 0.0003446136880557
Word: writ, Probability: 0.0003101523192501
Word: worse, Probability: 0.0002756909504446
Word: note, Probability: 0.0002067682128334
Word: late, Probability: 0.0002067682128334
Word: waste, Probability: 0.0001723068440278
Word: water, Probability: 0.0001723068440278
Word: wide, Probability: 0.0001723068440278
Word: wert, Probability: 0.0001378454752223
Word: wise, Probability: 0.0001378454752223
Word: wit, Probability: 0.0001033841064167
Word: woe, Probability: 0.0001033841064167
Word: urge, Probability: 0.0001033841064167
Word: white, Probability: 0.0001033841064167
Word: date, Probability: 0.0000689227376111
Word: writes, Probability: 0.0000689227376111
Word: rate, Probability: 0.0000689227376111
Word: wore, Probability: 0.0000689227376111
Word: free, Probability: 0.0000689227376111
Word: wine, Probability: 0.0000344613688056
Word: wits, Probability: 0.0000344613688056
Word: rite, Probability: 0.0000344613688056
```

همان طور که مشخص می باشد بیشترین احتمال برای کلمه art با احتمال ۰.۰۰۱۸۶۰۹۱۳۹۱۵۵۰۰۷ می باشد.