

Contents

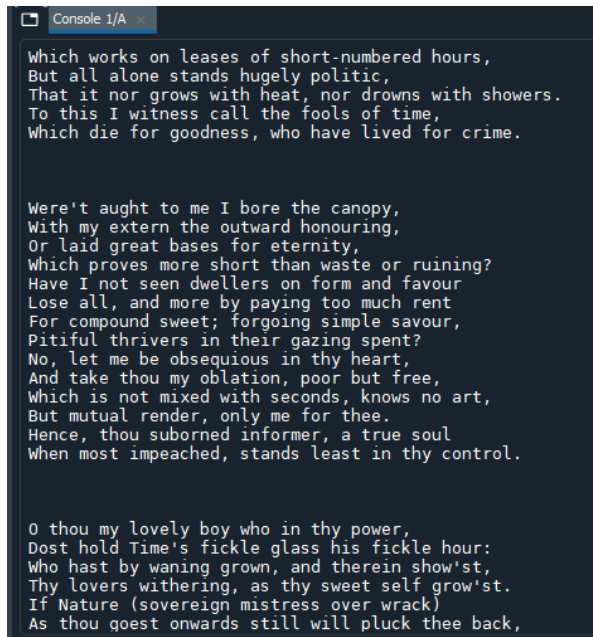
A. Importing files: open the file, Store the contents of the txt file in files	2
B. Text data preprocessing	2
a. Use regular expressions to restore abbreviations	2
b. Substitute.....	3
c. (Q1) Lowercase and remove punctuations and non-English words.	3
C. Split words	4
D.(Q1) Count the number of words in the article	5
E. (Q2) Calculate the frequency of the top 20 words in the file	5
F. (Q5) Data Visualization	6
a. Bar chart.....	6
b. Word Cloud	7
G. (Q3) Build a dictionary and arrange it in alphabetical order	8
a. Build a dictionary and sort by alphabetical order.....	8
b. Count the number of words in the dictionary	9
H. (Q4) Part-of-speech tagging	9
I: (Q4) N-gram tagging	10
a. I use brown_news in nltk to do the train the model.	10
b. Save the trained tagger, load it and use it, and finally get the part-of-speech tagging of three sentences.	11
c.(Q5) Next, visually analyze the content of part-of-speech tagging, and use a pie chart to display the verbs, nouns, pronouns, conjunctions, and adverbs.	11

Preliminary preparation:

A. Importing files: open the file, Store the contents of the txt file in files

```
#open the file
with open('C:\\Users\\USER\\Downloads\\Shakespeare.txt') as f:
    files= f.read()
    print(files)
```

Print the file, you will get part of the file as the following screenshot below:



Console 1/A

Which works on leases of short-numbered hours,
But all alone stands hugely politic,
That it nor grows with heat, nor drowns with showers.
To this I witness call the fools of time,
Which die for goodness, who have lived for crime.

Were't aught to me I bore the canopy,
With my extern the outward honouring,
Or laid great bases for eternity,
Which proves more short than waste or ruining?
Have I not seen dwellers on form and favour
Lose all, and more by paying too much rent
For compound sweet; forgoing simple savour,
Pitiful thrivers in their gazing spent?
No, let me be obsequious in thy heart,
And take thou my oblation, poor but free,
Which is not mixed with seconds, knows no art,
But mutual render, only me for thee.
Hence, thou suborned informer, a true soul
When most impeached, stands least in thy control.

O thou my lovely boy who in thy power,
Dost hold Time's fickle glass his fickle hour:
Who hast by waning grown, and therein show'st,
Thy lovers withering, as thy sweet self grow'st.
If Nature (sovereign mistress over wrack)
As thou goest onwards still will pluck thee back,

B. Text data preprocessing

When you open the document, you will find that there are many abbreviations and punctuation marks in the document. At this time, you need to restore these abbreviations and remove useless punctuation marks. The operation steps are as follows:

a. Use regular expressions to restore abbreviations

```
12 import re
13 pat_is = re.compile("(it|he|she|that|this|there|here)(\\'s)", re.I) # to
14 pat_s = re.compile("(?<=[a-zA-Z])\\'s")# to find the 's following the l
15 pat_s2 = re.compile("(?<=s)\\'s?")# to find the ' following the words e
16 pat_not = re.compile("(?<=[a-zA-Z])n\\'t")# to find the abbreviation of
17 pat_would = re.compile("(?<=[a-zA-Z])\\'d")# to find the abbreviation of
18 pat_will = re.compile("(?<=[a-zA-Z])\\'ll")# to find the abbreviation of
19 pat_am = re.compile("(?<=[Ii])\\'m")# to find the abbreviation of am
20 pat_are = re.compile("(?<=[a-zA-Z])\\'re")# to find the abbreviation of
21 pat_ve = re.compile("(?<=[a-zA-Z])\\'ve") # to find the abbreviation of
22 #After searching, it is found that there are some special abbreviations
23 pat_the = re.compile("(?<=[a-zA-Z]th/Th)(\\'s)") #th' means the
24 pat_verb = re.compile("(lov|mak|ceiv|ow|gav|deserv|rud)(\\'st)", re.I)#
```

As shown in line 12, I first load regular expression packages; As shown in line 13, I find the **'s following the pronouns**. re.I is refers to ignore case, such as it's; As shown in line 14, I find Possessive form, find the **'S following letters**, such as world's; As shown in line 15, I find the **' following the words ending by s**, such as others', Plural noun possessive; As shown in line 16, I find the **abbreviation of not**, like doesn't; As shown in line 17, I find the **abbreviation of would**, like I'd; As shown in line 18, I find the **abbreviation of will**, such as he'll; As shown in line 19, I find the **abbreviation of am**, like I'm; As shown in line 20, I find the **abbreviation of are**, like they're; As shown in line 21, I find the **abbreviation of have**, I've.

After that, I found some special abbreviations in the source file. As shown in line 23, I find th' & Th' means "the","The". As shown in line 24, I find that Shakespeare add " 'st " after the verb, which is a kind of grammar in medieval English, verbs in the second person are usually followed by 'st, which has no special meaning. The need for morphological changes has almost no effect on the meaning of the verb itself, so it is directly removed when restoring.

b. Substitute

```

26 #substitute
27 files = pat_is.sub(r"\l is", files)
28 #'s /s'represents the possessive form, which is directly removed when
29 files = pat_s.sub("", files)
30 files = pat_s2.sub("", files)
31 files = pat_not.sub(" not", files) # turn 't into not
32 files = pat_would.sub(" would", files) # turn 'd into would
33 files = pat_will.sub(" will", files) # turn 'll into will
34 files = pat_am.sub(" am", files) # turn 'm into am
35 files = pat_are.sub(" are", files) # turn 're into are
36 files = pat_ve.sub(" have", files) # turn 've into have
37 files = pat_the.sub("e ",files) # turn th' into the
38 files = pat_verb.sub(r"\le", files) # add e to after the verb less e
39 files = files.replace("'t"," it ") # Turned "'t" into "it"
40 files = files.replace("'Tis"," It is ") # turn 'Tis' into 'it is'
41 files = files.replace("'er","ver ") # turn 'er' into 'ver'
42 files = files.replace("'st","") # remove 'st
43 files = files.replace('\'', '') # remove '
44

```

As line 27 shows, I replace 's following the pronoun to is; As line 29,30 shows, 's indicates the possessive form, which is directly removed when restoring, and the verb prototype is retained; As line 31 shows, I Turned "'t" into "not" ; As line 32 shows, I turn 'd into would; As line 33 shows, I turn 'll into will; As line 34 shows, I turn 'm into am; As line 35 shows, I turn 're into are; As line 36 shows, I turn 've into have; As line 37 shows, I turn th' into the; As line 38 shows, I add e to after the verb less e; As line 39 shows, I Turned "'t" into "it"; As line 40 shows, I turn 'Tis' into 'it is'; As line 41 shows, I turn 'er' into 'ver'; As line 42 shows, I remove 'st. As line 43 shows, I remove '.

c. (Q1) Lowercase and remove punctuations and non-English words.

I don't consider lowercase and uppercase of words, uniformly converted to lowercase. Remove non-word characters.

```
46 files=files.lower()
47 #Filter out all non-word characters
48 noneng_word=re.compile(r"\W+")
49 files=re.sub(noneng_word," ",files)
50 #Complete file before split
51 print(files)
```

```
...: print(files)
the sonnets by william shakespeare from fairest creatures we desire increase that thereby beauty rose
might never die but as the ripper should by time decease his tender heir might bear his memory but thou
contracted to thine own bright eyes feedt thy light flame with self substantial fuel making a famine
where abundance lies thy self thy foe to thy sweet self too cruel thou that art now the world fresh
ornament and only herald to the gaudy spring within thine own bud buriest thy content and tender churl
makt waste in niggarding pity the world or else this glutton be to eat the world due by the grave and
thee when forty winters shall besiege thy brow and dig deep trenches in thy beauty field thy youth proud
livery so gazed on now will be a tattered weed of small worth held then being asked where all thy beauty
lies where all the treasure of thy lusty days to say within thine own deep sunken eyes were an all
eating shame and thriftless praise how much more praise deserved thy beauty use if thou couldst answer
this fair child of mine shall sum my count and make my old excusee proving his beauty by succession
thine this were to be new made when thou art old and see thy blood warm when thou felt it cold look in
thy glass and tell the face thou viewest now is the time that face should form another whose fresh
repair if now thou not renewest thou dost beguile the world unbless some mother for where is she so fair
whose unearned womb disdains the tillage of thy husbandry or who is he so fond will be the tomb of his
self love to stop posterity thou art thy mother glass and she in thee calls back the lovely april of her
prime so thou through windows of thine age shalt see despite of wrinkles this thy golden time but if
thou live remembered not to be die single and thine image dies with thee unthrifty loveliness why dost
thou spend upon thy self thy beauty legacy nature bequest gives nothing but doth lend and being frank
she lends to those are free then beauteous niggard why dost thou abuse the bounteous largess given thee
to give profitless usurer why dost thou use so great a sum of sums yet canst not live for having traffic
with thy self alone thou of thy self thy sweet self dost deceive then how when nature calls thee to be
gone what acceptable audit canst thou leave thy unused beauty must be tombed with thee which used lives
the executor to be those hours that with gentle work did frame the lovely gaze where every eye doth
dwell will play the tyrants to the very same and that unfair which fairly doth excel for never resting
time leads summer on to hideous winter and confounds him there ear checked with frost and lusty leaves
```

C. Split words

```
55 words=files.split()
56 print(words)
```

```

....: print(words)
['the', 'sonnets', 'by', 'william', 'shakespeare', 'from', 'fairest', 'creatures', 'we', 'desire',
'increase', 'that', 'thereby', 'beauty', 'rose', 'might', 'never', 'die', 'but', 'as', 'the', 'riper',
'should', 'by', 'time', 'decease', 'his', 'tender', 'heir', 'might', 'bear', 'his', 'memory', 'but',
'thou', 'contracted', 'to', 'thine', 'own', 'bright', 'eyes', 'feedt', 'thy', 'light', 'flame', 'with',
'self', 'substantial', 'fuel', 'making', 'a', 'famine', 'where', 'abundance', 'lies', 'thy', 'self',
'thy', 'foe', 'to', 'thy', 'sweet', 'self', 'too', 'cruel', 'thou', 'that', 'art', 'now', 'the',
'world', 'fresh', 'ornament', 'and', 'only', 'herald', 'to', 'the', 'gaudy', 'spring', 'within',
'thine', 'own', 'bud', 'buried', 'thy', 'content', 'and', 'tender', 'churl', 'makt', 'waste', 'in',
'niggarding', 'pity', 'the', 'world', 'or', 'else', 'this', 'glutton', 'be', 'to', 'eat', 'the',
'world', 'due', 'by', 'the', 'grave', 'and', 'thee', 'when', 'forty', 'winters', 'shall', 'besiege',
'thy', 'brow', 'and', 'dig', 'deep', 'trenches', 'in', 'thy', 'beauty', 'field', 'thy', 'youth',
'proud', 'livery', 'so', 'gazed', 'on', 'now', 'will', 'be', 'a', 'tattered', 'weed', 'of', 'small',
'worth', 'held', 'then', 'being', 'asked', 'where', 'all', 'thy', 'beauty', 'lies', 'where', 'all',
'the', 'treasure', 'of', 'thy', 'lusty', 'days', 'to', 'say', 'within', 'thine', 'own', 'deep',
'sunken', 'eyes', 'were', 'an', 'all', 'eating', 'shame', 'and', 'thrifless', 'praise', 'how', 'much',
'more', 'praise', 'deserved', 'thy', 'beauty', 'use', 'if', 'thou', 'couldst', 'answer', 'this', 'fair',
'child', 'of', 'mine', 'shall', 'sum', 'my', 'count', 'and', 'make', 'my', 'old', 'excusee', 'proving',
'his', 'beauty', 'by', 'succession', 'thine', 'this', 'were', 'to', 'be', 'new', 'made', 'when', 'thou',
'art', 'old', 'and', 'see', 'thy', 'blood', 'warm', 'when', 'thou', 'felt', 'it', 'cold', 'look', 'in',
'thy', 'glass', 'and', 'tell', 'the', 'face', 'thou', 'viewest', 'now', 'is', 'the', 'time', 'that',
'face', 'should', 'form', 'another', 'whose', 'fresh', 'repair', 'if', 'now', 'thou', 'not', 'renewest',
'thou', 'dost', 'beguile', 'the', 'world', 'unbless', 'some', 'mother', 'for', 'where', 'is', 'she',
'so', 'fair', 'whose', 'unearned', 'womb', 'disdains', 'the', 'tillage', 'of', 'thy', 'husbandry', 'or',
'who', 'is', 'he', 'so', 'fond', 'will', 'be', 'the', 'tomb', 'of', 'his', 'self', 'love', 'to', 'stop',
'posterity', 'thou', 'art', 'thy', 'mother', 'glass', 'and', 'she', 'in', 'thee', 'calls', 'back',
'the', 'lovely', 'april', 'of', 'her', 'prime', 'so', 'thou', 'through', 'windows', 'of', 'thine',
'age', 'shalt', 'see', 'despite', 'of', 'wrinkles', 'this', 'thy', 'golden', 'time', 'but', 'if',
'thou', 'live', 'remembered', 'not', 'to', 'be', 'die', 'single', 'and', 'thine', 'image', 'dies',
'with', 'thee', 'unthrifty', 'loveliness', 'why', 'dost', 'thou', 'spend', 'upon', 'thy', 'self', 'thy',
'beauty', 'legacy', 'nature', 'bequest', 'gives', 'nothing', 'but', 'doth', 'lend', 'and', 'being']

```

D.(Q1) Count the number of words in the article

Circulate each word in “words, which is word after split”, if it is a word, count+1

```

60     count=0
61     # for loop word in the collection after split
62     for word in words:
63         # if it is a word, count+1
64         count+= len(words)
65     print("The number of words in the whole article is:",count)

```

count the number of words after split, get the result:

```

....: print("The number of words in the whole article is:",count)
The number of words in the whole article is: 17706

```

E. (Q2) Calculate the frequency of the top 20 words in the file

I use Series in pandas to calculate the frequency of top 20 words.

```

69     import pandas as pd
70     fre=pd.Series(words).value_counts()
71     print("The frequency of the top 20 words are:")
72     print(fre[:20])

```

```

...: print(fre[:20])
The frequency of the top 20 words are:
and      490
the      441
to       414
my       393
of       370
i        351
that     323
in       323
thy      287
thou     235
love     194
is       184
with     181
for      172
not      167
a        164
me       164
but      163
thee     162
so       145
dtype: int64

In [6]:

```

As the figure shows, in the top 20 words, “and” appears 490 times. Some high-frequency words are words that are not helpful for text analysis, so the keywords with effective information will be displayed in the word cloud later.

F. (Q5) Data Visualization

a. Bar chart

I use matplotlib package to do data visualization for all the words and non-repetitive words in the file.

```

77 import matplotlib.pyplot as plt
78 #count the unique word
79 unique_count=0
80 for word in fre:
81     unique_count+= len(word)
82 #print(unique_count)
83 # count is total number of words; un
84 x=("count","unique_count")
85 y=(count,unique_count)
86 plt.bar(x,y)
87 plt.title('frequency')
88 plt.show()

```

```

...: print(unique_count)
3099

```

In the file, non-repetitive words appears 3099 times. There are two abscissas of the bar graph, namely count and unique_count, and y is the value of both of them. The count value is 17706 and the unique_count value is 3099.

Word cloud is to visually highlight keywords that appear frequently in the text through color and size. In the graph I can see some high frequency keywords such as: love, thee, thy, doth, thou, will...

G. (Q3) Build a dictionary and arrange it in alphabetical order

a. Build a dictionary and sort by alphabetical order

First, I define the function to build dictionary, I calculate the frequency of words and sort them by alphabetical order. And then build dictionary based on words. The 112th, I name the unknown words as "unk".

```
104 import pandas as pd
105 # define a dictionary function, in which we have two parameters: one is w
106 def build_dict(sp_words, min_word_freq=0):
107     word_freq = pd.Series(sp_words).value_counts() #show frequency of e
108     word_freq = filter(lambda x: x[1]>=min_word_freq, word_freq.items())
109     word_freq_sorted = sorted(word_freq, key=lambda x: (x[0], x[1]))
110     words, _ = list(zip(*word_freq_sorted)) # output the result as a
111     word_idx = dict(zip(words, range(len(words)))) # The dict() func
112     word_idx['<unk>'] = len(words) #unk means unknown, unknown word
113     return word_idx
114 # call the build_dict function.word is words after split, 1 is True.
115 dic=build_dict(words,1)
116 print("The content of dictionary:",dic)
```

And then I call the function as line 115 and get the result:

As the below picture shows, the former is a word, the latter is a serial number.


```

...: print("The content of dictionary:",dic)
The content of dictionary: {'a': 0, 'abhor': 1, 'abide': 2, 'able': 3,
'about': 4, 'above': 5, 'absence': 6, 'absent': 7, 'abundance': 8,
'abundant': 9, 'abuse': 10, 'abused': 11, 'abuses': 12, 'abysm': 13,
'accents': 14, 'acceptable': 15, 'acceptance': 16, 'accessary': 17,
'accident': 18, 'accidents': 19, 'account': 20, 'accumulate': 21, 'accuse':
22, 'accusing': 23, 'achieve': 24, 'acknowledge': 25, 'acquaintance': 26,
'acquainted': 27, 'act': 28, 'action': 29, 'active': 30, 'actor': 31, 'add':
32, 'added': 33, 'adder': 34, 'addeth': 35, 'adding': 36, 'addition': 37,
'adieu': 38, 'adjunct': 39, 'admire': 40, 'admired': 41, 'admiring': 42,
'admit': 43, 'admitted': 44, 'adonis': 45, 'adore': 46, 'adulterate': 47,
'advance': 48, 'advantage': 49, 'adverse': 50, 'advised': 51, 'advocate':
52, 'afar': 53, 'affable': 54, 'affairs': 55, 'affections': 56, 'afford':
57, 'affords': 58, 'afloat': 59, 'afresh': 60, 'after': 61, 'afterwards':
62, 'again': 63, 'against': 64, 'age': 65, 'ages': 66, 'aggravate': 67,
'ah': 68, 'aid': 69, 'air': 70, 'alack': 71, 'alas': 72, 'alchemy': 73,
'alien': 74, 'alike': 75, 'alive': 76, 'all': 77, 'allayed': 78, 'allege':
79, 'allow': 80, 'almost': 81, 'aloft': 82, 'alone': 83, 'already': 84,
'alter': 85, 'alteration': 86, 'altered': 87, 'alters': 88, 'although': 89,
'altring': 90, 'always': 91, 'am': 92, 'amazeth': 93, 'ambush': 94, 'amen':
95, 'amends': 96, 'amis': 97, 'amiss': 98, 'among': 99, 'an': 100,
'anchored': 101, 'and': 102, 'anew': 103, 'angel': 104, 'anger': 105,
'angry': 106, 'annexed': 107, 'annoy': 108, 'anon': 109, 'another': 110,
'answer': 111, 'answered': 112, 'answers': 113, 'anticipate': 114,
'antique': 115, 'antiquity': 116, 'any': 117, 'apparel': 118, 'appeal': 119,
'appear': 120, 'appearance': 121, 'appearing': 122, 'appears': 123,
'appetite': 124, 'apple': 125, 'applying': 126, 'approve': 127, 'april':
128, 'are': 129, 'arest': 130, 'argument': 131, 'aright': 132, 'arise': 133,
'arising': 134, 'array': 135, 'arrest': 136, 'art': 137, 'arts': 138, 'as':
139, 'ashes': 140, 'aside': 141, 'askance': 142, 'asked': 143, 'asleep':
144, 'aspect': 145, 'assailed': 146, 'assemble': 147, 'assistance': 148

```

b. Count the number of words in the dictionary

```

120     dic_c=0
121     for word in dic:
122         dic_c+=len(dic)
123     print("The number of words in the dictionary is: ",dic_c)

```

The result is as follows:

```

...: print("The number of words in the dictionary is: ",dic_c)
The number of words in the dictionary is: 3100

```

H. (Q4) Part-of-speech tagging

I use nltk package and then select three sentences in the article, t1,t2,t3. Then I build a corpus for these three sentences. And then I use pos_tag in nltk to do the pos tagging for the words in the three sentences.

```

131 import nltk
132 t1="You had a father, let your son say so."
133 t2="That Time will come and take my love away "
134 t3="Nature bequest gives nothing but doth lend. "
135 # put t1,t2,t3 into a corpus, named dict
136 dict= list(set(nltk.word_tokenize(t1)+nltk.word_tokenize(t2)+nltk.word_tokenize(t3)))
137 # use pos_tag in nltk to do part of speech tagging.
138 tag=nltk.pos_tag(dict)
139 print(tag)

```

The results are as follows:

```

...: print(tag)
[('son', 'NN'), ('but', 'CC'), ('father', 'RB'), ('away', 'RB'), ('lend', 'VB'), ('JJ'), ('my', 'PRP$'), ('will', 'MD'), ('your', 'PRP$'), ('love', 'VB'), ('so', 'RB'), ('a', 'DT'), ('come', 'NN'), (',', ','), ('take', 'VB'), ('doth', 'NN'), ('.', '.'), ('Time', 'NNP'), ('had', 'VBD'), ('Nature', 'NNP'), ('You', 'PRP'), ('nothing', 'NN'), ('That', 'WDT'), ('gives', 'VBZ'), ('bequest', 'RBS'), ('let', 'NNS'), ('say', 'VB'), ('and', 'CC')]
In [12]:

```

Pos tagging can do simple pos tagging, but the accuracy rate is not very high, for example, in the three sentences, father appears as NN, but the result I get is RB. Therefore, I use n-gram.

I: (Q4) N-gram tagging

a. I use brown_news in nltk to do the train the model.

First, I use the news category in brown_news in nltk for training. Among them, train_sets is the sentences used in training, and 90% of all sentences are used as the training set. test_sets is the test set, and the remaining sentences are used. There is a sparse problem in n-gram, that is, when n is larger, the specificity of the context will increase, and the probability that the data to be labeled contains context that does not exist in the training data also increases, so a combined tagger is required. Proceed as follows: 1 Use Unigram tagger to tag identifiers; 2 If Bigram tagger cannot find the tag, try Unigram tagger; 3. If the Trigram tagger cannot find the tag, try the Bigram tagger; 4. If the Trigram tagger cannot find the tag, the default tagger is used.

```

142 from nltk.corpus import brown
143 from nltk.tag import UnigramTagger
144 from nltk.tag import BigramTagger
145 from nltk.tag import TrigramTagger
146 #brown news as a training set training model
147 brown_tagged_sents = brown.tagged_sents(categories='news')
148 default_tagger = nltk.DefaultTagger('NN')
149 #training set
150 train_data=brown_tagged_sents[:int(len(brown_tagged_sents)*0.9)]
151 test_data=brown_tagged_sents[int(len(brown_tagged_sents)*0.9):]
152 #Unigram
153 unigram_tagger=UnigramTagger(train_data,backoff=default_tagger)
154 print("The result of unigram is:",unigram_tagger.evaluate(test_data))
155 #Bigram
156 bigram_tagger=BigramTagger(train_data,backoff=unigram_tagger)
157 print("The result of bigram is:",bigram_tagger.evaluate(test_data))
158 #Trigram
159 trigram_tagger=TrigramTagger(train_data,backoff=bigram_tagger)
160 print("The result of trigram is:",trigram_tagger.evaluate(test_data))

```

The final success rate after training is shown in the figure below:

```

The result of unigram is: 0.8361407355726104
The result of bigram is: 0.8452108043456593
The result of trigram is: 0.843317053722715

```

b. Save the trained tagger, load it and use it, and finally get the part-of-speech tagging of three sentences.

```

163 from pickle import dump
164 output = open('unigram_tagger.pkl','wb')
165 dump(unigram_tagger,output,-1)
166 output.close()
167 #Load tagger
168 from pickle import load
169 input = open('unigram_tagger.pkl','rb')
170 tagger = load(input)
171 input.close()
172 #Use tagger
173 #part of speech tagging for three sentences above
174 tagged=tagger.tag(dict)
175 print("The tags of three sentences are:",tagged)

```

```

...: print("The tags of three sentences are:",tagged)
The tags of three sentences are: [('son', 'NN'), ('but', 'CC'), ('father', 'NN'), ('away', 'RB'),
('lend', 'VB'), ('my', 'PP$'), ('will', 'MD'), ('your', 'PP$'), ('love', 'VB'), ('so', 'QL'), ('a',
'AT'), ('come', 'VB'), ('', ''), ('take', 'VB'), ('doth', 'NN'), ('.', '.'), ('Time', 'NN-TL'),
('had', 'HVD'), ('Nature', 'NN'), ('You', 'PPSS'), ('nothing', 'PN'), ('That', 'DT'), ('gives', 'VBZ'),
('bequest', 'NN'), ('let', 'VB'), ('say', 'VB'), ('and', 'CC')]

```

In [20]:

My result is correct, as you can see, the father has become NN.

c.(Q5) Next, visually analyze the content of part-of-speech tagging, and use a pie chart to display the verbs, nouns, pronouns, conjunctions, and adverbs.

```

178 all_noun=[word for word ,pos in tagged if pos in ['NN','NN-TL']]
179 all_verb=[word for word ,pos in tagged if pos in ['VB','HVD','VBN','MD']]
180 all_pron=[word for word ,pos in tagged if pos in ['DT','PPSS','PP$']]
181 all_adv=[word for word ,pos in tagged if pos in ['RB']]
182 all_conj=[word for word ,pos in tagged if pos in ['CC']]
183 p0=len(dict)
184 p1=len(all_noun)/p0
185 p2=len(all_verb)/p0
186 p3=len(all_pron)/p0
187 p4=len(all_adv)/p0
188 p5=len(all_conj)/p0
189 p6=1-p1-p2-p3-p4-p5
190 labels=['noun','verb','pron','adv','conj','others']
191 X=[p1,p2,p3,p4,p5,p6]
192 colors = ["#5bae23", "#dfecd5", "#cad3c3", "#9fa39a", "#b2cf87", "#96c24e"]
193 fig = plt.figure()
194 # Draw a pie chart (data, the label corresponding to the data, color, and the
195 plt.pie(X, labels=labels, colors=colors, autopct='%1.2f%%')
196 plt.title("Tag Pie chart")
197 plt.show()

```

The following results are obtained: Among them, verb accounts for 29.63% of the total part-of-speech tags; Among them, noun accounts for 22.22% of the total part-of-speech tags; Among them, others accounted for 22.22% of the total part-of-speech tags; Among them, pron accounts for 14.81% of the total part-of-speech tags; Among them, conj accounts for 7.41% of the total part-of-speech tags; Among them, adverb accounts for 3.7% of the total part-of-speech tags.

