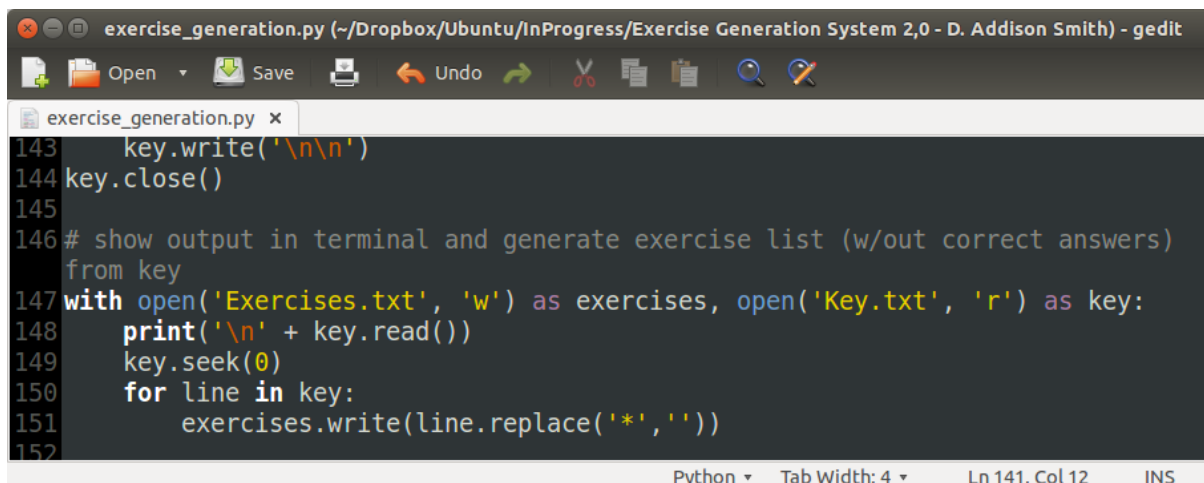


Automatic phrasal verb exercise generation for learners of English

This system has been designed and implemented with the objective *to test English language learners on their knowledge of phrasal verbs and associated particles and prepositions*. The program takes as input a text file from which to pull the example sentences, and so the desired level of difficulty should be represented there. Alternatively, if no input file is passed as a command line argument, the system randomly intakes one hundred (tagged) Brown corpus sentences of the *news* and *government* categories. The Natural Language ToolKit (NLTK) and WordNet are used among the current 150 or so lines of Python code (see below and appendix), which therefore means the accuracy of the system is related partly to that of the various tools such as Part of Speech tagger, for example.

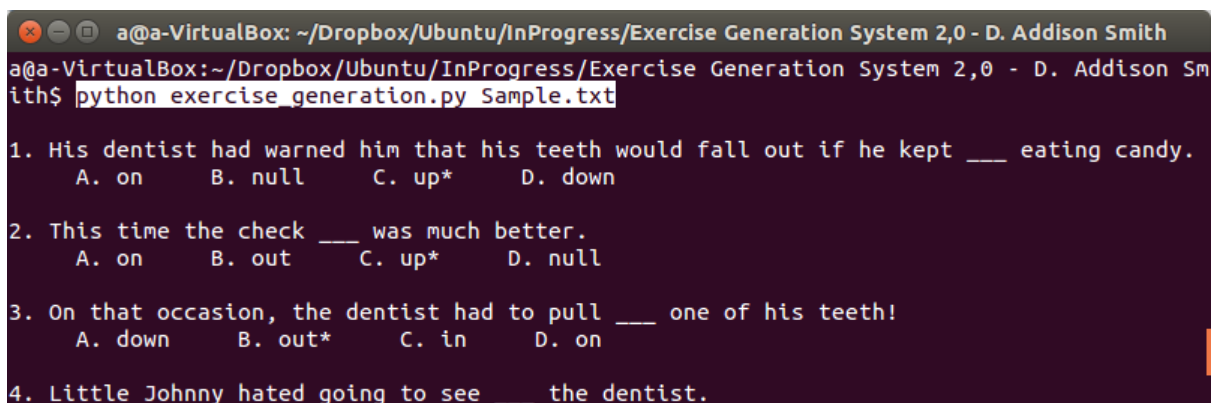


```

exercise_generation.py (~/.Dropbox/Ubuntu/InProgress/Exercise Generation System 2,0 - D. Addison Smith) - gedit
Open Save Undo Redo
exercise_generation.py x
143 key.write('\n\n')
144 key.close()
145
146 # show output in terminal and generate exercise list (w/out correct answers)
147 from key
148 with open('Exercises.txt', 'w') as exercises, open('Key.txt', 'r') as key:
149     print('\n' + key.read())
150     key.seek(0)
151     for line in key:
152         exercises.write(line.replace('*', ''))
Python Tab Width: 4 Ln 141, Col 12 INS

```

First, the textual input (if provided as below) is separated into sentences using a sentence tokenizer from the mentioned NLTK toolkit. Each sentence is then itself tokenized and Part of Speech tags are added to the individual words.

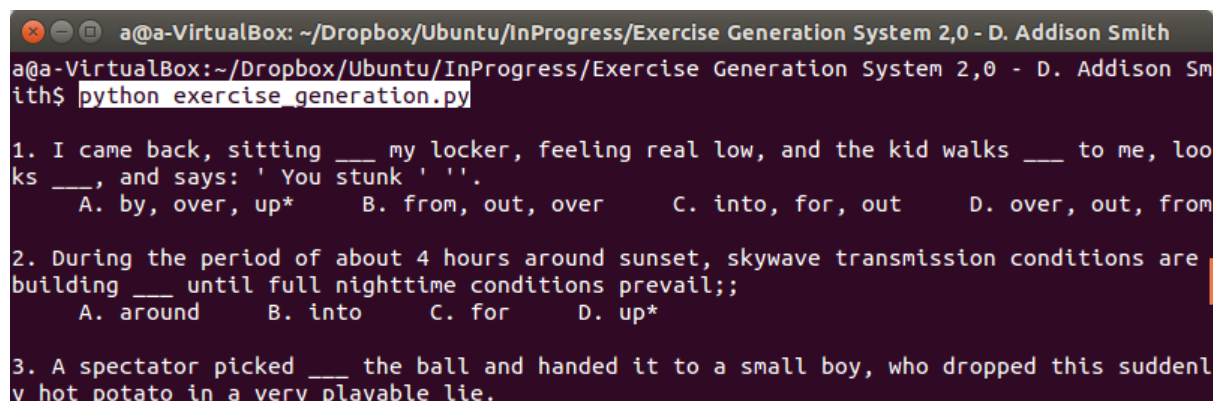


```

a@a-VirtualBox: ~/Dropbox/Ubuntu/InProgress/Exercise Generation System 2,0 - D. Addison Smith
a@a-VirtualBox:~/Dropbox/Ubuntu/InProgress/Exercise Generation System 2,0 - D. Addison Smith$ python exercise_generation.py Sample.txt
1. His dentist had warned him that his teeth would fall out if he kept ___ eating candy.
   A. on      B. null   C. up*     D. down
2. This time the check ___ was much better.
   A. on      B. out    C. up*     D. null
3. On that occasion, the dentist had to pull ___ one of his teeth!
   A. down    B. out*   C. in      D. on
4. Little Johnny hated going to see ___ the dentist.

```

Alternatively, if no input text is provided, the program randomly grabs one hundred tagged Brown corpus sentences as mentioned (example below).



```
a@a-VirtualBox: ~/Dropbox/Ubuntu/InProgress/Exercise Generation System 2,0 - D. Addison Smith
a@a-VirtualBox:~/Dropbox/Ubuntu/InProgress/Exercise Generation System 2,0 - D. Addison Smith$ python exercise_generation.py

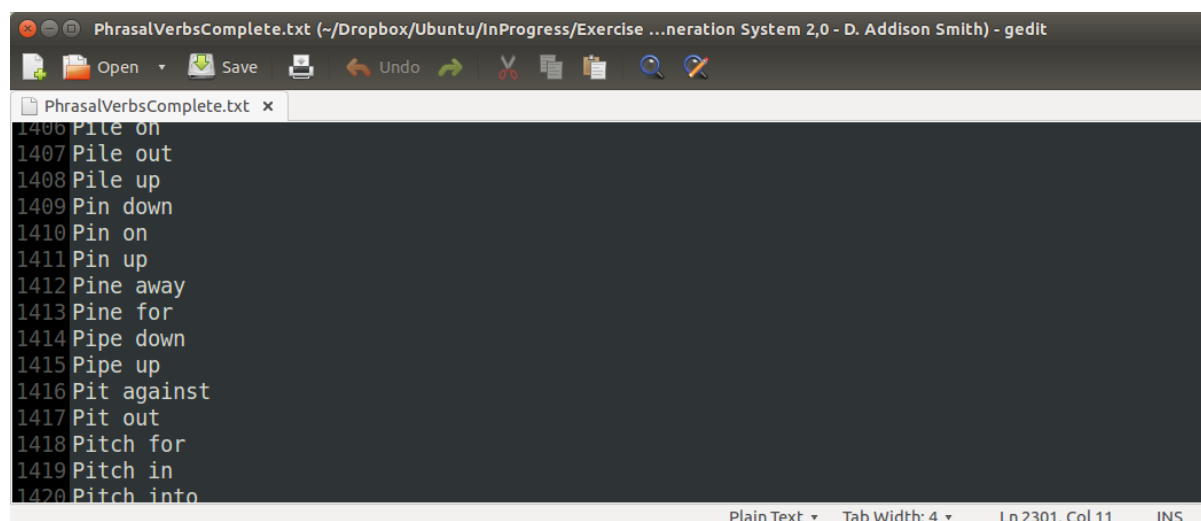
1. I came back, sitting ____ my locker, feeling real low, and the kid walks ____ to me, looks ____, and says: ' You stunk ' '.
   A. by, over, up*    B. from, out, over    C. into, for, out    D. over, out, from

2. During the period of about 4 hours around sunset, skywave transmission conditions are building ____ until full nighttime conditions prevail;;
   A. around    B. into    C. for    D. up*

3. A spectator picked ____ the ball and handed it to a small boy, who dropped this suddenly hot potato in a very playable lie.
```

A list of particles and prepositions used in the input sentences is extracted to later make up distractor choices, and sentences that do *not* contain phrasal verbs are discarded with one exception: phrases that meet certain criteria are kept as candidates for the null option of the program, if enabled.

Extracted prepositional (phrasal) verb occurrences are compared against a stored list of possible phrasal verbs¹ (see below) for verification. Particle and prepositional verbs are pulled from simple contexts, and in the case that a given phrasal verb has both an RP and IN, only the RP blank is provided at this point. Lemmatized verbs are stored, and possible phrasal verb combinations are looked up via WordNet. These possible particles and prepositions are added as potential distractors, but with only one such addition coming from each synset in order to (attempt to) limit ambiguity.



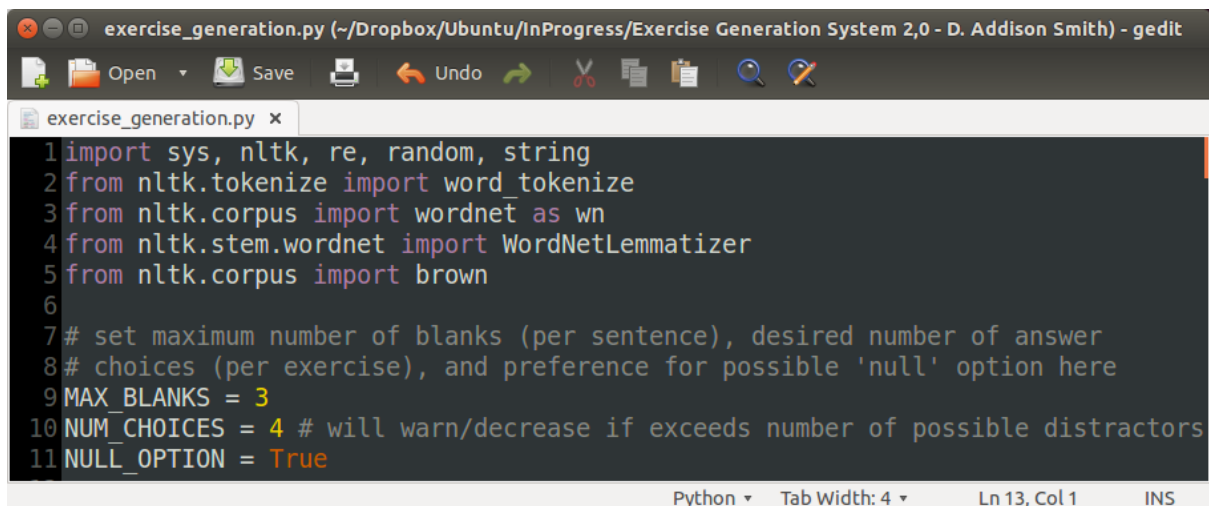
```
PhrasalVerbsComplete.txt (-~/Dropbox/Ubuntu/InProgress/Exercise ...neration System 2,0 - D. Addison Smith) - gedit

1406 Pile on
1407 Pile out
1408 Pile up
1409 Pin down
1410 Pin on
1411 Pin up
1412 Pine away
1413 Pine for
1414 Pipe down
1415 Pipe up
1416 Pit against
1417 Pit out
1418 Pitch for
1419 Pitch in
1420 Pitch into
```

¹ Acquired from <http://www.usingenglish.com/reference/phrasal-verbs/list.html>

An output file is then created and the remaining sentences are printed as individual exercises using blanks for particles, prepositions, and possible null instances. The program is able to handle sentences with multiple phrasal verbs and/or null instances, and such a sentence is represented as a single exercise (creating multiple exercises from a single sentence would give away the answers to other exercises generated from the same sentence).

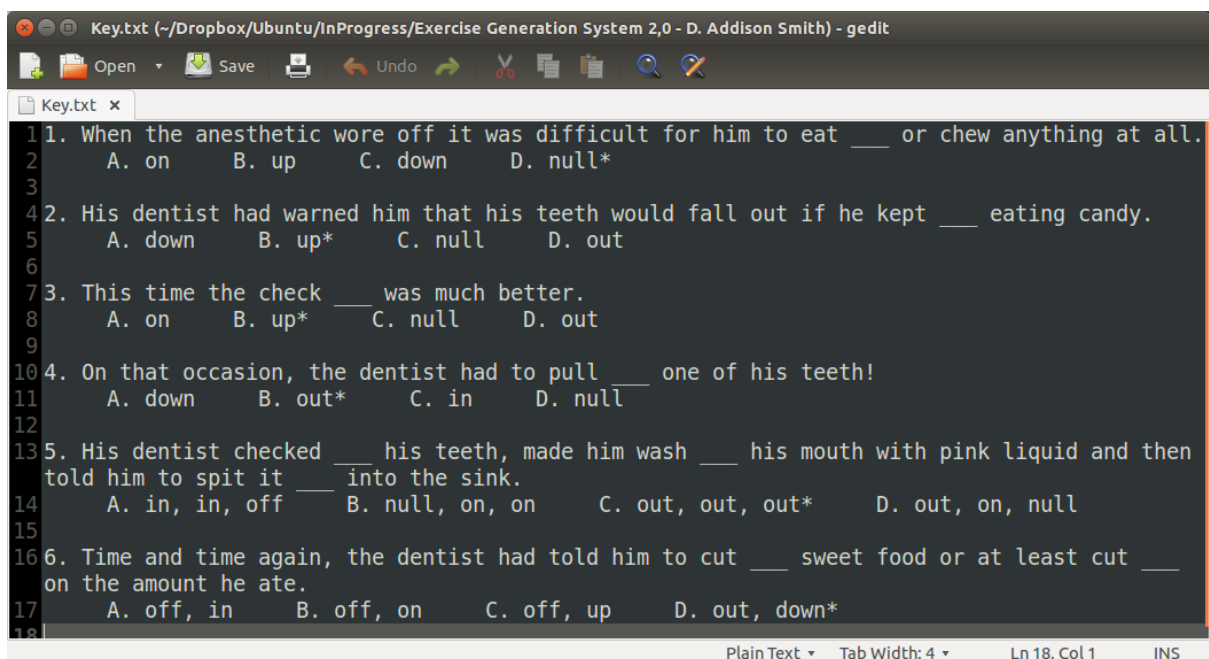
A global variable NUM_CHOICES (see below) represents the desired number of answer choices (including the key) for each exercise, and so is easily changeable. MAX_BLANKS allows the user to specify the maximum number of desired blanks per exercise, after which the rest of the sentence is simply printed in its entirety. Last of the global variables is the Boolean NULL_OPTION, which allows the optional addition of the *null* distractor. Enabling this feature also randomly adds one to three correct instances of the *null* option (where verbs are used in non-phrasal contexts) relative to the number of phrasal verb exercises. In other words, the *maximum* number of correct null instances is proportional to the number of extracted sentences containing phrasal verbs.



```
1 import sys, nltk, re, random, string
2 from nltk.tokenize import word_tokenize
3 from nltk.corpus import wordnet as wn
4 from nltk.stem.wordnet import WordNetLemmatizer
5 from nltk.corpus import brown
6
7 # set maximum number of blanks (per sentence), desired number of answer
8 # choices (per exercise), and preference for possible 'null' option here
9 MAX_BLANKS = 3
10 NUM_CHOICES = 4 # will warn/decrease if exceeds number of possible distractors
11 NULL_OPTION = True
```

Distractors are randomly generated from the set of all particles and prepositions used in the input text along with possible *null* option, and then the answer choices are displayed in random order. In the case of a multi-blank sentence, distractors are generated with number of particles and prepositions to match the sentence (i.e. if a single sentence has three blanks, three-word distractors are created in order to mimic the key). The correct answer choice, at the moment, is marked with an asterisk (*).

Question numbers and answer-choice letters are displayed, and the program outputs an answer key (Key.txt), a text file of exercises without correct responses (Exercises.txt), and also to the console (see below). These respective screenshots demonstrate the types of output along with the differences in the randomly produced answer choices, both in the choices themselves and also their respective order. The ability to cope with sentences with multiple phrasal verbs and/or null instances can also be observed. Here the number of desired answer choices is set to four, although this can currently be changed to any positive integer given that it does not exceed the number of gathered particles and prepositions—plus one (*null* option), if enabled.



```
Key.txt (-/Dropbox/Ubuntu/InProgress/Exercise Generation System 2,0 - D. Addison Smith) - gedit
1 1. When the anesthetic wore off it was difficult for him to eat ___ or chew anything at all.
2   A. on    B. up    C. down   D. null*
3
4 2. His dentist had warned him that his teeth would fall out if he kept ___ eating candy.
5   A. down   B. up*   C. null   D. out
6
7 3. This time the check ___ was much better.
8   A. on     B. up*   C. null   D. out
9
10 4. On that occasion, the dentist had to pull ___ one of his teeth!
11  A. down   B. out*  C. in     D. null
12
13 5. His dentist checked ___ his teeth, made him wash ___ his mouth with pink liquid and then
14   told him to spit it ___ into the sink.
15   A. in, in, off   B. null, on, on   C. out, out, out*   D. out, on, null
16
17 6. Time and time again, the dentist had told him to cut ___ sweet food or at least cut ___
18   on the amount he ate.
19   A. off, in    B. off, on    C. off, up    D. out, down*
```

```
Exercises.txt (-/Dropbox/Ubuntu/InProgress/Exercise Generation System 2,0 - D. Addison Smith) - gedit
1 1. His dentist had warned him that his teeth would fall out if he kept ___ eating candy.
2   A. out    B. down    C. up    D. on
3
4 2. This time the check ___ was much better.
5   A. down    B. up    C. null    D. out
6
7 3. Johnny was delighted ___ and so was his dentist.
8   A. up    B. null    C. down    D. on
9
10 4. On that occasion, the dentist had to pull ___ one of his teeth!
11   A. up    B. on    C. out    D. down
12
13 5. His dentist checked ___ his teeth, made him wash ___ his mouth with pink liquid and then
14   told him to spit it ___ into the sink.
15   A. up, down, down    B. up, off, off    C. up, null, up    D. out, out, out
16
17 6. Time and time again, the dentist had told him to cut ___ sweet food or at least cut ___
18   on the amount he ate.
   A. off, off    B. up, down    C. out, down    D. down, up
Plain Text ▾ Tab Width: 4 ▾ Ln 1, Col 1 INS
```

```
a@a-VirtualBox: ~/Dropbox/Ubuntu/InProgress/Exercise Generation System 2,0 - D. Addison Smith
a@a-VirtualBox:~/Dropbox/Ubuntu/InProgress/Exercise Generation System 2,0 - D. Addison Smith$ python exercise_genera
tion.py Sample.txt
1. When the anesthetic wore off it was difficult for him to eat ___ or chew anything at all.
   A. on    B. down    C. null*    D. out
2. His dentist had warned him that his teeth would fall out if he kept ___ eating candy.
   A. down    B. up*    C. on    D. null
3. This time the check ___ was much better.
   A. out    B. null    C. down    D. up*
4. On that occasion, the dentist had to pull ___ one of his teeth!
   A. up    B. down    C. out*    D. on
5. His dentist checked ___ his teeth, made him wash ___ his mouth with pink liquid and then told him to spit it ___
into the sink.
   A. up, up, out    B. out, out, out*    C. off, down, up    D. on, in, null
6. Time and time again, the dentist had told him to cut ___ sweet food or at least cut ___ on the amount he ate.
   A. up, null    B. up, on    C. on, out    D. out, down*
```

A type of exception handling has been added in order to cope with instances where the globally set number of answer choices per exercise exceeds the number of possible choices for a given exercise. In this case, a warning is printed to the console and the number of choices is automatically reduced only for the specific exercise, after which operation continues as normal (see below). This is the exact same scenario as earlier screenshot but this time without the *null* option, which in turn causes one exercise not to have enough distractors to support the desired four answer choices.

```
a@a-VirtualBox: ~/Dropbox/Ubuntu/InProgress/Exercise Generation System 2,0 - D. Addison Smith
a@a-VirtualBox:~/Dropbox/Ubuntu/InProgress/Exercise Generation System 2,0 - D. Addison Smith$ python exercise_genera
tion.py Sample.txt

NUM_CHOICES exceeds number of possible distractors;
automatically decreased to 3 for Exercise 1.

1. This time the check ___ was much better.
   A. out   B. up*  C. down

2. His dentist had warned him that his teeth would fall out if he kept ___ eating candy.
   A. on    B. out  C. down  D. up*

3. On that occasion, the dentist had to pull ___ one of his teeth!
   A. out*  B. in   C. up    D. down

4. His dentist checked ___ his teeth, made him wash ___ his mouth with pink liquid and then told him to spit it ___
   into the sink.
   A. out, out, out*  B. off, on, down  C. off, down, out  D. up, out, up

5. Time and time again, the dentist had told him to cut ___ sweet food or at least cut ___ on the amount he ate.
   A. up, up  B. down, up  C. down, on  D. out, down*
```

This project was initially presented on May 13th, 2016 as part of the Education and NLP portion of the Advanced Applications in Language Technologies course at University of the Basque Country, and is now presented as the final project of same course.



Automatic phrasal verb exercise generation for learners of English

D. Addison Smith

June, 2016

Advanced Applications in Language Technologies: Education and NLP



The system in its current state is somewhat vulnerable to cases where two different answer choices might both be correct in a given instance, and so future work should include research into further disambiguation.

Appendix

```
1. import sys, nltk, re, random, string
2. from nltk.tokenize import word_tokenize
3. from nltk.corpus import wordnet as wn
4. from nltk.stem.wordnet import WordNetLemmatizer
5. from nltk.corpus import brown
6.
7. # set maximum number of blanks (per sentence), desired number of answer
8. # choices (per exercise), and preference for possible 'null' option here
9. MAX_BLANKS = 3
10. NUM_CHOICES = 4 # will warn/decrease if exceeds number of possible distractors
11. NULL_OPTION = True
12.
13. tagged_sents = []
14. # if provided, import file passed as parameter, split sentences, and tag Parts of Speech
15. if len(sys.argv) > 1:
16.     with open(sys.argv[1], 'r') as my_file:
17.         sents = nltk.sent_tokenize(my_file.read())
18.         for sent in sents:
19.             tagged_sents.append(nltk.pos_tag(word_tokenize(sent)))
20. # else randomly intake 100 (tagged) Brown corpus sentences of certain categories
21. else:
22.     tagged_sents = random.sample(brown.tagged_sents(categories=['news', 'government']), 100)
23.
24. rpins = set()
25. rpin_sents = set()
26. in_occurrences = []
27. null_candidates = []
28. selected_null_candidates = []
29. lmtzr = WordNetLemmatizer()
30.
31. # add 'null' option if global Boolean set to True
32. if NULL_OPTION:
33.     rpins.add('null')
34.
35. for sent in tagged_sents:
36.     # import phrasal verb list acquired from http://www.usingenglish.com/reference/phrasal-verbs/list.html
37.     f = open('PhrasalVerbsComplete.txt', 'r')
38.     for w, word in enumerate(sent):
39.         # store phrasal verb particles (RPs) and respective sentences
40.         if word[1] == 'RP' and w != 0:
41.             rpins.add(word[0])
42.             rpin_sents.add(tuple(sent))
43.         # store phrasal verb prepositions (INs) and respective sentences
44.         elif re.match(r'VB.*', word[1]) and len(sent) > w+1 and sent[w+1][1] == 'IN':
45.             # valid occurrence if lemmatized verb and IN combination appear in list of possible phrasal verbs
46.             for line in (x for x in f if x.lower().strip() == lmtzr.lemmatize(word[0], wn.VERB) + " " + sent[w+1][0]):
47.                 # keep track of prepositional verb sentences and indices of occurrences
48.                 in_occurrences.append([tuple(sent), w])
49.                 rpins.add(sent[w+1][0])
50.                 rpin_sents.add(tuple(sent))
51.         # store phrase candidates for null option if enabled globally
52.         elif re.match(r'VB.*', word[1]) and NULL_OPTION:
53.             found_cand = False
54.             phrase = word[0] + " "
55.             wordcount = 1
56.             # scan the rest of current sentence
```



```

57.         for i in range(w+1, len(sent)):
58.             # keep up with current phrase candidate
59.             phrase += sent[i][0] + " "
60.             # increment word count only if non-punctuation
61.             if not re.match(r'\.|\,|\!|\?|\;|\:', sent[i][0]):
62.                 wordcount += 1
63.             # weed out poor candidates for null option
64.             if (re.match(r'VB.*|PRP|JJ|RB', sent[i][1]) and i == w+1) or re.matc
h(r'RP|IN|TO', sent[i][1]):
65.                 break
66.             # if reach another verb or end of sentence, found null candidate
67.             elif re.match(r'VB.*', sent[i][1]) or i == len(sent)-1:
68.                 found_cand = True
69.                 break
70.             # further scrutinize null candidates by setting minimum number of words
71.
72.         if found_cand and wordcount > 2:
73.             # keep track of null phrase candidate(s) and indices of locations
74.             null_candidates.append([tuple(sent),w])
75.     f.close()
76. # randomly select one to three null candidates, taking into consideration number of
77. # phrasal verb occurrences
78. if null_candidates and NULL_OPTION:
79.     selected_null_candidates = random.sample(null_candidates, random.sample(xrange(1,
80. min((len(rpin_sents)//10)+2,4)),1)[0])
81.     for null_candidate in selected_null_candidates:
82.         rpin_sents.add(tuple(null_candidate[0]))
83. # generate key (exercises and correct answers), writing sentences and leaving RP/IN/
84. # null blank(s)
85. key = open('Key.txt', 'w')
86. for x, rp_sent in enumerate(rpin_sents):
87.     correct = ''
88.     numBlanks = 0
89.     verbs = set()
90.     # write exercise numbers
91.     key.write(str(x+1) + '. ')
92.     for i, word in enumerate(rp_sent):
93.         # do not write space before punctuation
94.         if re.match(r'\.|\,|\!|\?|\;|\:', word[0]):
95.             key.write(word[0])
96.             elif numBlanks < MAX_BLANKS and ((word[1] == 'RP' and i != 0) or (word[1] ==
97. 'IN' and [rp_sent,i-1] in in_occurrences)):
98.                 numBlanks += 1
99.                 key.write(' ____')
100.                 correct += word[0] + ', '
101.                 elif numBlanks < MAX_BLANKS and NULL_OPTION and [rp_sent,i] in selected_null
102. _candidates:
103.                     numBlanks += 1
104.                     key.write(' ' + word[0] + ' ____')
105.                     correct += 'null, '
106.                 else:
107.                     key.write(' ' + word[0])
108.             # store lemmatized verbs to look up possible phrasal verb combos for distra
109. ctor generation
110.             if re.match(r'VB.*', word[1]):
111.                 verbs.add(lmtzr.lemmatize(word[0], wn.VERB))
112.             key.write('\n')
113. # add possible RP/IN options for stored verbs from a given sentence via WordNet
114.
115. for verb in verbs:
116.     for synset in wn.synsets(verb, pos=wn.VERB):
117.         for lemma in (x for x in synset.lemmas() if '_' in x.name()):
118.             lem = nltk.pos_tag(word_tokenize(lemma.name().replace('_', ' ')))

```



```

114.         if verb == lem[0][0] and re.match(r'RP|IN', lem[1][1]):
115.             rpins.add(lem[1][0])
116.             # lower ambiguity by accepting only one phrasal verb per synset
117.
118.             break
119.
120.     # initialize set of answer choices with correct choice
121.     choices = {correct}
122.     # randomly create and write answer choices
123.     num_choices = NUM_CHOICES
124.     while len(choices) < num_choices:
125.         # warn/decrease NUM_CHOICES if exceeds number of possible distractors for a
n exercise
126.         if num_choices > len(rpins) * numBlanks:
127.             while num_choices > len(rpins) * numBlanks:
128.                 num_choices -= 1
129.                 print "\nNUM_CHOICES exceeds number of possible distractors;"
130.                 print "automatically decreased to " + str(num_choices) + " for Exercise
" + str(x+1) + "."
131.                 choice = ''
132.                 # create and add random distractor for given number of blanks
133.                 for i in range(numBlanks):
134.                     choice += str(random.sample(rpins,1)[0]) + ', '
135.                 choices.add(choice)
136.                 choiceNum = 0
137.                 # write choice letters and distractors in random order
138.                 for choice in random.sample(choices, num_choices):
139.                     key.write(' ' + string.ascii_uppercase[choiceNum] + ". " + choice[:-
2])
140.                     # indicate correct choice in printout
141.                     if choice == correct:
142.                         key.write('*')
143.                         choiceNum += 1
144.                     key.write('\n\n')
145. key.close()
146. # show output in terminal and generate exercise list (w/out correct answers) from k
ey
147. with open('Exercises.txt', 'w') as exercises, open('Key.txt', 'r') as key:
148.     print('\n' + key.read())
149.     key.seek(0)
150.     for line in key:
151.         exercises.write(line.replace('*', ''))

```