EDAN20 - LAB 0

Language Technology EDAN20 @ LTH - http://cs.lth.se/edan20/coursework/laboratory-0/ (http://cs.lth.se/edan20/coursework/laboratory-0/)

Author Jonatan Kronander

Imports and setup

In [1]:
```
import re
from collections import Counter

def words(text): return re.findall(r'\w+', text.lower())
WORDS = Counter(words(open('big.txt').read()))
```

How many words in corpus?

In [23]:
```
len(WORDS)
```

Out[23]: 32198

Which is the most common words?

In [25]:
```
WORDS.most_common(10)
```

Out[25]:
```
[('the', 79809),
 ('of', 40024),
 ('and', 38312),
 ('to', 28765),
 ('in', 22023),
 ('a', 21124),
 ('that', 12512),
 ('he', 12401),
 ('was', 11410),
 ('it', 10681)]
```

1. Selection Mechanism: argmax

We suggest the word that is most probable. Using max('argmax') in python.

2.Candidate Model: c ∈ candidates

This model tells us what other words to concider

In [2]:
```
def edits1(word):
    "All edits that are one edit away from `word`."
    letters    = 'abcdefghijklmnopqrstuvwxyz'
    splits     = [(word[:i], word[i:])    for i in range(len(word) + 1)] #Split to two words?
    deletes    = [L + R[1:]               for L, R in splits if R] #Remove one letter
    transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1] #Swap two adjacent letters
    replaces   = [L + c + R[1:]           for L, R in splits if R for c in letters] #Change one letter to another
    inserts    = [L + c + R               for L, R in splits for c in letters] #Add a letter
    return set(deletes + transposes + replaces + inserts)
```

This can be a big set. For a word of length n, there will be n deletions, n-1 transpositions, 26n alterations, and 26(n+1) insertions, for a total of 54n+25 (of which a few are typically duplicates). For example,

In [15]:
```
len(edits1('helllo'))
```

Out[15]: 334

We want to only suggest the words that is in the corpus. (in big.txt)

In [4]:
```
def known(words): return set(w for w in words if w in WORDS)
```

In [16]:
```
known(edits1('helllo'))
```

Out[16]: {'hello'}

Maybe two edits can also be usefull. This creates a bigger suggestion set.

In [6]:
```
def edits2(word): return (e2 for e1 in edits1(word) for e2 in edits1(e1))
```

In [17]:
```
len(set(edits2('helllo')))
```

Out[17]: 49232

Also here we only want to suggest the words that is in the corpus. (in big.txt)

In [18]:
```
known(edits2('helllo'))
```

Out[18]: {'hallo', 'hell', 'hello', 'hullo'}

3.Language Model: P(c)

The probability that c appears as a word of English text. For example, occurrences of "the" make up about 7% of English text, so we should have P(the) = 0.07.

We use the function below to estimate the probability for a word in the coprus

In [26]:
```
def P(word, N=sum(WORDS.values()), W = WORDS): return W[word] / N
```

In [27]:
```
P('the')
```

Out[27]: 0.07154004401278254

In [30]:
```
P('hello')
```

Out[30]: 8.963906829152417e-07

In [31]:
```
P('notincorpus')
```

Out[31]: 0.0

4. Error Model: P(w|c)

The probability that w would be typed in a text when the author meant c. For example, P(teh|the) is relatively high, but P(theeexyz|the) would be very low.

In [32]:
```
def correction(word): return max(candidates(word), key=P) # Most probable spelling correction for word.

def candidates(word):
    # Generate possible spelling corrections for word.
    return known([word]) or known(edits1(word)) or known(edits2(word)) or [word]
```

Test with own report

Lets check my latest report I wrote in Translational Neuromodeling course at ETH.

In [33]:
```
WORDS_test = Counter(words(open('test.txt').read()))
```

How many words?

In [34]:
```
len(WORDS_test)
```

Out[34]: 762

Which is the most common words in my report?

In [35]:
```
WORDS_test.most_common(10)
```

Out[35]:
```
[('the', 149),
 ('a', 75),
 ('in', 66),
 ('agent', 62),
 ('to', 62),
 ('and', 48),
 ('learning', 47),
 ('of', 43),
 ('reward', 36),
 ('is', 30)]
```

How frequent is 'the' used?

In [36]:
```
P('the', sum(WORDS_test.values()), WORDS_test)
```

Out[36]: 0.05788655788655789

Which words was "wrongly" spelled? Using the big.txt as corpus.

In [37]:
```
WORDS_wrong = list(WORDS_test)

for word in list(WORDS_test):
    if word in list(WORDS):
        WORDS_wrong.remove(word)
```

Which words was spelled wrong the most?

In [38]:
```
dict_wrong = {}

for word in WORDS_wrong:
    i = WORDS_wrong.index(word)
    dict_wrong[word] = P(word, sum(WORDS_test.values()), WORDS_test)
```

In [39]:
```
dict(Counter(dict_wrong).most_common(10))
```

Out[39]:
```
{'maze': 0.006604506604506605,
 'optimal': 0.006216006216006216,
 'simulations': 0.004662004662004662,
 'parameters': 0.0034965034965034965,
 'rl': 0.003108003108003108,
 'mdp': 0.002331002331002331,
 'algorithm': 0.0019425019425019425,
 'agentwith': 0.001554001554001554,
 'learningbehaviour': 0.001554001554001554,
 'theagent': 0.001554001554001554}
```

What suggestions do we have for our wrongly spelled words? Using the big.txt as corpus.

In [40]:
```
WORDS_suggestions = list()
for word in WORDS_wrong:
    WORDS_suggestions.append(known(edits2(word)))
```

In [41]:
```
WORDS_suggestions[WORDS_wrong.index('optimal')]
```

Out[41]: {'optical', 'optional'}

In [42]:
```
WORDS_suggestions[WORDS_wrong.index('parameters')]
```

Out[42]: {'parameter'}

We can see that our corpus is probably not good enough...

In [43]:
```
correction('optimal')
```

Out[43]: 'optical'

In [44]:
```
candidates('optimal')
```

Out[44]: {'optical'}