Section 1:

N-gram Language Modeling

In [2]:
```
from nltk.corpus import brown
from nltk.tokenize import word_tokenize
```

In [8]:
```
# Loading the corpus
corpus = brown.words()

# Case folding and getting vocab
lower_case_corpus = [w.lower() for w in corpus]
vocab = set(lower_case_corpus)

print('CORPUS EXAMPLE: ' + str(lower_case_corpus[:30]) + '\n\n')
print('VOCAB EXAMPLE: ' + str(list(vocab)[:10]))
```

```
CORPUS EXAMPLE: ['the', 'fulton', 'county', 'grand', 'jury', 'said', 'friday', 'an', 'investigation', 'of', "atlanta's", 'recent', 'primary', 'electio
n', 'produced', '``', 'no', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'place', '.', 'the', 'jury', 'further', 'said', 'in']


VOCAB EXAMPLE: ['drudgery', 'one-arm', 'growling', 'cutest', 'rain', 'hops', "network's", 'expressionists', 'polarization', 'gaussian']
```

In [9]:
```
print('Total words in Corpus: ' + str(len(lower_case_corpus)))
print('Vocab of the Corpus: ' + str(len(vocab)))
```

```
Total words in Corpus: 1161192
Vocab of the Corpus: 49815
```

In [15]:
```
bigram_counts = {}
trigram_counts = {}

# Sliding through corpus to get bigram and trigram counts
for i in range(len(lower_case_corpus) - 2):
    # Getting bigram and trigram at each slide
    bigram = (lower_case_corpus[i], lower_case_corpus[i+1])
    trigram = (lower_case_corpus[i], lower_case_corpus[i+1], lower_case_corpus[i+2])

    # Keeping track of the bigram counts
    if bigram in bigram_counts.keys():
        bigram_counts[bigram] += 1
    else:
        bigram_counts[bigram] = 1

    # Keeping track of trigram counts
    if trigram in trigram_counts.keys():
        trigram_counts[trigram] += 1
    else:
        trigram_counts[trigram] = 1

print("Example, count for bigram ('the', 'king') is: " + str(bigram_counts[('the', 'king')]))
```

```
Example, count for bigram ('the', 'king') is: 51
```

In [4]:
```
# Function takes sentence as input and suggests possible words that comes after the sentence
def suggest_next_word(input_, bigram_counts, trigram_counts, vocab):
    # Consider the last bigram of sentence
    tokenized_input = word_tokenize(input_.lower())
    last_bigram = tokenized_input[-2:]

    # Calculating probability for each word in vocab
    vocab_probabilities = {}
    for vocab_word in vocab:
        test_trigram = (last_bigram[0], last_bigram[1], vocab_word)
        test_bigram = (last_bigram[0], last_bigram[1])

        test_trigram_count = trigram_counts.get(test_trigram, 0)
        test_bigram_count = bigram_counts.get(test_bigram, 0)

        probability = test_trigram_count / test_bigram_count
        vocab_probabilities[vocab_word] = probability

    # Sorting the vocab probability in descending order to get top probable words
    top_suggestions = sorted(vocab_probabilities.items(), key=lambda x: x[1], reverse=True)[:3]
    return top_suggestions
```

In [5]:
```
suggest_next_word('I am the king', bigram_counts, trigram_counts, vocab)
```

Out[5]:
```
[('james', 0.17647058823529413),
 ('of', 0.1568627450980392),
 ('arthur', 0.11764705882352941)]
```

In [6]:
```
suggest_next_word('I am the king of', bigram_counts, trigram_counts, vocab)
```

Out[6]:
```
[('france', 0.3333333333333333),
 ('hearts', 0.16666666666666666),
 ('morocco', 0.08333333333333333)]
```

In [7]:
```
suggest_next_word('I am the king of france', bigram_counts, trigram_counts, vocab)
```

Out[7]:
```
[('and', 0.26666666666666666), ('.', 0.26666666666666666), (',', 0.2)]
```

In [9]:
```
suggest_next_word('I am the king of france and', bigram_counts, trigram_counts, vocab)
```

Out[9]:
```
[('the', 0.2), ('germany', 0.13333333333333333), ('some', 0.06666666666666667)]
```

Section 2:

Use N-gram language model with Laplace smoothing and sentence generation.

Data preprocessing in train.txt and test.txt files should already be processed such that:

1. Punctuation is removed
2. Each sentence is on its own line

Example output for a trigram model trained on data/train.txt and tested against data/test.txt:

```
Loading 3-gram model...
Vocabulary size: 23505
Generating sentences...
...
<s> <s> the company said it has agreed to sell its shares in a statement </s>
(0.03163)
<s> <s> he said the company also announced measures to boost its domestic economy and
could be a long term debt </s> (0.01418)
<s> <s> this is a major trade bill that would be the first quarter of 1987 </s>
(0.02182)
...
Model perplexity: 51.555
```