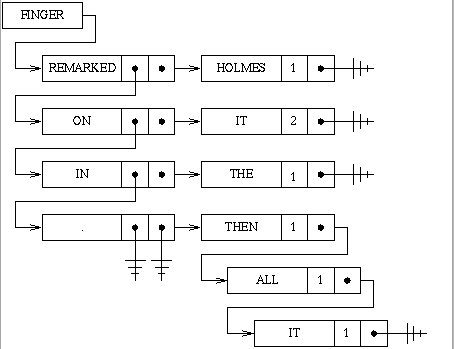
In this assignment, you will fit a tri-gram language model to English and then use it to generate new english text.

A unigram model of English consists of a single probability distribution ***P****(W)* over the set of all words.

A bigram model of English consists of two probability distributions: ***P****(W0)* and ***P****(Wi | Wi-1)*. The first distribution is just the probability of the first word in a document. The second distribution is the probability of seeing word *Wi* given that the previous word was *Wi-1*.

A trigram model of English consists of three probability distributions: ***P****(W0)*, ***P****(W1|W0)*, and ***P****(Wi|Wi-1,Wi-2)*. The first distribution is, as above, the probability of the first word in the document. The next distribution is the probability of the second word given the first one. And the third distribution is the probability of the *i*th word given the two preceding words.

Given a set of documents, your job in this assignment is to fit a trigram model of English. I recommend that you do this by using a hash table in which you hash on word Wi-2. The contents of the hash table cells consist of linked lists as shown below. Each item in the main list links the words that appeared at position Wi-1. It also contains a pointer to a second level of linked lists that link the words that appeared at position Wi.



In particular, this structure encodes the fact that in our training data, we observed the following three word sequences:

finger remarked holmes

finger on it

finger on it

finger in the

finger . then

finger . all

finger . it

Notice that "finger on it" was observed twice. Also notice that the period is treated as a separate word.

Given the information in this data structure, we can compute the probability *P(it | finger, on)* as 2/2 = 1. Similarly, we can compute the probability *P(it | finger, .)* as 1/3.

Using the given corpus, train a tri-gram language model by constructing the hash/linked list data structure described above. Then use this data structure to generate a new "story" 1000 words long. You can do this very simply by first choosing a word at random from the hash table. Then using it to choose a subsequent word, and then extending the text by looking up the two words and choosing at random from among the following words in proportion to their frequency of appearance.

Repeat this process, but now train on all six books and then generate a new "story" of 1000 words .