ContextObjectDescription

1. **A simpler mission**.
The Opinion Mining project has, ostensibly, succeeded in streamlining the procedure in this case. In that partcular undertaking, we managed to construct a few algorithms that can be used here almost as-is. Among these is:
1) The algorithm to parse a text file and create a list of words
2) An algorithm to apply insertion sort to strings
3) Linear searching in a file based on an ordered list in RAM

Out of these, the first algorithm must be modified to count occurences of words in a list.

2. **Procedure**.
The overall plan is as:

⟨ 1. Start. ⟩
⟨ 2. Prepare pointers to word-list and to file. ⟩
⟨ 3. Read words from file and record frequency of occurrence. ⟩
⟨ 4. Sort words by frequency of occurence. ⟩
⟨ 5. Extract nouns. ⟩
⟨ 6. Identify frequent nouns based on cutoff. ⟩
⟨ 7. Stop. ⟩

At the risk of being repititive, we will record all algorithms that are identical below, as in the Opinion Mining project- for reference. This program will have one more argument- the *cutoff* percentage- which will act as a threshold to qualify a noun as *frequent*. Step (2) is pretty straightforward, obviously- but this may not be so in other implementations.

⟨ 2. Prepare pointers to word-list and to file. ⟩:-
char *word_list* **[1000][1000];**

```
FILE *F ← fopen( argv[1], "r");
```

Step (3) has to be suitably modified, as indicated.

```
⟨ 3. Read words from file and record frequency of occurrence.
⟩:-
word_no ← 0;

while( ~ feof( F ) )
{

  fscanf( F, "%s", &temp );

  (Remove punctuation at the end of the word.)
  c ← temp [ strlen( temp ) - 1 ];
  if( ispunct( c ) )
    temp [ strlen( temp ) - 1 ] ← '\0' ;

  (Record frequency of occurence.)
  repeated ← 0;
  for( i ← 0; i ≤ word_no; i ← i + 1 )
    if( strcmp ( temp, word_list [ i ] ) = 0 )
    {
      freq [ i ] ← freq [ i ] + 1;
      repeated ← 1;
    }

  if( repeated = 0 )
  {
    strcpy ( word_list [ word_no ], temp );
    freq [ word_no ] ← 1;
    word_no ← word_no + 1;
  }

}
```

For the sorting step, we can reuse the linear search algorithm for strings, with the frequency used to index words rather than their lexicographic values. This means that the "swap" step in linear searching invloves the swapping of items in *freq* as well as words in *word_list*.

```
⟨ 4. Sort words by frequency of occurence. ⟩:-
word_count ← word_no;

i ← 1;
while( i < word_count )
{
  j ← i;

  while( (j > 0) ∧ (freq [ j - 1 ] > freq [ j ]) )
  {
    (Swap:)
    strcpy( temp, word_list [ j ]);
    strcpy( word_list [ j ], word_list [ j - 1 ]);
    strcpy( word_list [ j - 1 ], temp);
    t ← freq [ j ];
    freq [ j ] ← freq [ j - 1 ];
    freq [ j - 1 ] ← t ;

    j ← j - 1;
  }

  i ← i + 1;
}
```

The circumstance when the control leaves this block is as follows; the file has been read completely, *word_no* has a count of the words in the list, *freq* contains a record of the number of times each word occurs in the input statement. The step (4) also differs somewhat significantly from the similar step in Opinion Mining, in that yet another array in RAM, the *Nouns* array must be formed.

```
⟨ 5. Extract nouns. ⟩:-
word_no ← 0; i ← 0;

for( word_no ← 0; word_no < word_count ; word_no ← word_no
+ 1 )
{

   strcpy ( T, word_list [ word_no ] );

   (Linear Search:)
   fscanf ( Noun_file, "%s", &noun );

   while( ~feof ( Noun_file ) )
   {
     if( strcmp ( noun, T ) = 0 )
     { strcpy ( Nouns [ i ], T ); i ← i + 1; }
     fscanf ( Noun_file, "%s", &noun );
   }

   (Start again at top of file.)
   fseek ( Noun_file, 0, SEEK_SET );

}
```

We will allow for the cutoff to be effectively zero percent by rounding *down* to the value of the last "frequent" item in *Nouns*.

```
⟨ 6. Identify frequent nouns based on cutoff. ⟩:-

noun_count ← i;

cutoff ← atoi ( argv [3] ) (Read integer valued argument- cutoff.) ;
for(
   j ← 0;
   j ≤ (int)( cutoff × noun_count / 100);
```

```
    j ← j + 1
    )
    printf ( "\n%s", Nouns [ noun_count - 1 - j ] );
```

Finally,

```
⟨ 1. Start. ⟩:-
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>

int main ( int argc , char *argv [] )
{

  int i, j, t, word_no, word_count, noun_count, cutoff, repeated,
freq [1000];
  char c, noun [1000], T [1000], temp [1000], Nouns [1000][1000];

  FILE *Noun_file ← fopen ( argv [2], "r" );
```

(Main program)

```
⟨ 7. Stop. ⟩:-

return 0;
}
```