COMPILING CONTEXTUALIZED LISTS OF FREQUENT VOCABULARY FROM USER-SUPPLIED CORPORA USING NATURAL LANGUAGE PROCESSING TECHNIQUES

BY

OMID ABDAR

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Arts in Teaching English as a Second Language
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Master’s Committee:

Associate Professor Randall W Sadler, Chair
Assistant Professor Lane Oscar Bingaman Schwartz
ABSTRACT

Since there are thousands of words to learn in a new language, one common challenge for language learners and teachers is knowing which vocabulary items to prioritize over the others and, in general, setting vocabulary-learning goals. Within vocabulary teaching research, one approach has been to focus on lists of the most common vocabulary. West (1953) proposed a list of the 2000 most frequent word families in English that, it was argued, were most important for learners to master. Along the same lines, Coxhead (2000) offered a list of the most common words in academic English known as the Academic Word List (AWL). Arguing that AWL did not adequately reflect the learners’ specialized vocabulary needs, however, corpus linguists began to develop wordlists in specialized subject areas with an English for Specific Purposes (ESP) perspective for students in Business, Engineering, Medical, and Law majors and so on. A central theme in almost all previous endeavors to develop better wordlists has been the notion of ‘representativeness’—the extent to which a wordlist ‘represents’ the language needs of leaners. In this study, it is proposed that an alternative way to maximize representativeness in a wordlist is to enable users to compile a wordlist from any text or corpus that is of interest to them and to provide the means of compiling a wordlist using that text. Using Natural Language Toolkit (NLTK), this study shows how a few Natural Language Processing (NLP) techniques may be used to compile a list of the most common words in the Europarl corpus along with retrieving example sentences from the corpus for each word. This new approach can have applications for both language leaners as well as for the purposes of preparing instructional materials in an ESP setting.
ACKNOWLEDGEMENTS

In the fall of 2015, I was taking a class called “Tools for Big Data” with Professor Schwartz that introduced me to some fundamental concepts in computer programming with Python and Natural Language Processing which inspired me to explore this area in more depth for my thesis. I owe Professor Schwartz many thanks for his patience with my countless questions that semester as well as later on and throughout the completion of this thesis, but more than anything I feel indebted to him for having given me the confidence to pursue research in an area where I had little background when I started. I would also like to thank Professor Sadler for his input and valuable feedback on this thesis. Needless to say, any errors great or small that remain are entirely my own.
## TABLE OF CONTENTS

- **CHAPTER 1: INTRODUCTION**.................................................................1
- **CHAPTER 2: METHODOLOGY**.............................................................8
- **CHAPTER 3: RESULTS**......................................................................14
- **CHAPTER 4: CONCLUSION**...............................................................20
- **REFERENCES**...................................................................................26
- **APPENDIX A: PROGRAM SCRIPT FOR PYTHON 2.7.0**......................27
CHAPTER 1
INTRODUCTION

Learning new vocabulary is among the most challenging areas of learning a new language. When students learn new vocabulary in a language, they need to become familiar with both the form and the many meanings that a certain word might have in different contexts, along with what language structures the word is typically used in and so on. It is due to these challenges that developing instructional materials that target the vocabulary needs of language learners has always been an active area of second language teaching research. Since there are hundreds of thousands of words to learn in a new language—and since they cannot all be learnt at the same time—it is important for learners and teachers alike to know what words to prioritize over the others. One popular way to start is by focusing on the most common words in a language. Learning this core vocabulary would offer the learners a foundation on which to add more vocabulary as they advance to higher levels of proficiency. While not all scholars agree on the effectiveness of using wordlists as a means of facilitating the learning and teaching of new vocabulary, a considerable amount of endeavor has been made in the past to provide lists of the most frequent words in a language in general, or within certain registers of a language. In these endeavors, knowledge of vocabulary has traditionally been categorized in the three main domains of ‘general service’, ‘academic’ and ‘technical.’

1.1 THE GENERAL SERVICE DOMAIN

The general service domain is associated with the language skills that fulfill the learners’ ordinary, day-to-day communication needs. The wordlists in this category are compiled by a comparison of numerous bodies of text and by identifying the items that occur most frequently
across these texts. Among the many different lists of frequent words available for English, by far
the most influential and widely used is West’s General Service List (GSL), first introduced in
1952, which contains the most common 2000 word families in English. GSL influenced both
pedagogy and vocabulary research and directly shaped the way essential English vocabulary was
conceptualized at the time.

While the significance of West’s GSL is well established in the literature, a number of problems
have been pointed out since it was introduced. An important critique of GSL is that, given how
many years ago it was compiled, it is arguably out of date since language is always changing.
GSL includes words that are almost barely used anymore such as cart, shilling, servant, footman,
milkmaid, and telegraph but fails to include others that one would expect to see in a list of
common words in English, such as television and computer (Richards 1974; Nation 1990). GSL
has also been criticized for being inconsistent in its selection of words. Richards (1974), for
example, pointed out that in the semantic field of animals, GSL includes items like bear,
elephant, and monkey; but excludes others such as lion, tiger, and fox. All critiques aside, GSL
founded the basis of modern corpus-based wordlists and it was even later shown that the core of
the GSL overlaps to a large extent with wordlists that were compiled years later using modern
corpus-based methods (Gilner & Morales, 2008). Other than compiling principles, wordlists
differ from each other in regards to what corpora they have been generated from. The most
notable of these corpora include the Lancaster-Oslo-Bergen Corpus (LOB), the British National
Corpus (BNC), the BE06 Corpus of British English (BE06), and EnTenTen12. LOB, BE06, and
EnTenTen12 represent written language only while BNC includes data from speech as well. The
most notable difference between these corpora, however, is their size—ranging from 1 million
words in LOB to more than 12 billion words in EnTenTen12. Table 1 provides more information
about these corpora. LOB and BE06 represent a wide range of written genres in English, including newspapers, fiction, essays, and scientific writing. BNC, which was compiled in the early 1990s, represents a mid-size corpus by today’s standards and is a balanced sample of British English that includes a substantial spoken part as well.

Table 1. Different corpora used to make wordlists in the general service domain.

<table>
<thead>
<tr>
<th></th>
<th>LOB</th>
<th>BNC</th>
<th>BE06</th>
<th>EnTenTen12</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Words</td>
<td>1 million</td>
<td>100 million</td>
<td>1 million</td>
<td>12 billion</td>
</tr>
<tr>
<td>Period</td>
<td>1961</td>
<td>1990s</td>
<td>2005-7</td>
<td>2012</td>
</tr>
<tr>
<td>No. of texts</td>
<td>500</td>
<td>4,049</td>
<td>500</td>
<td>1.55 million</td>
</tr>
</tbody>
</table>

EnTenTen12 is by far the largest of the four corpora, whose representativeness lies in its enormous size and coverage of a wide variety of online texts. Compiling wordlists in the general service domain was also greatly influential in writing Learner's dictionaries as it made it possible for these dictionaries to write definitions of all the words in the English language using only the highly frequent words, which in turn meant that learners could now consult a monolingual dictionary to look up the meaning of newly encountered words in English. Apart from having inspired the building of larger corpora and more comprehensive wordlists in the general service domain, GSL is also considered as having established the foundation of the more recent Academic Word List, which is concerned not with the general domain but rather Academic Writing in English.
1.2 THE ACADEMIC DOMAIN

As mentioned earlier, the recurring theme underlying any effort to compile a list of common vocabulary in any domain is the idea that identifying which words are common in a domain can help learners and teachers to set vocabulary learning goals. In other words, since there is a very large number of words to be learned, it makes most sense to focus on—or start with—what is most common. Since the primary purpose of many learners of English is to be able to study in academic programs in which English is the language of instruction, it is not surprising that many scholars have tried to develop a selection of ‘academic’ words that such students would need to learn. The earliest efforts in this regard culminated in the work of Campion and Elley (1971). The authors selected 19 different subject areas and by comparing various texts in these areas focused on words that occurred in at least 3 or more of the subject areas in order to identify the words that were common across all subject areas. Lynn (1973) looked more closely at what the learners thought was challenging language. They believed that a better way to compile a list of common academic terms that could be used by learners of English would be to manually examine which words the learners highlighted in their textbooks as challenging. They essentially argued that a list of words that previous learners had marked as ‘difficult’ would equal a list of words that future learners would naturally want to learn too.

The most recognized attempt to compile a list of common words in academic English was introduced by Coxhead (2000). Largely influenced by the GSL, Coxhead examined a corpus of 3.5 million running words in terms of range and frequency, in an effort to generate a list of words that appeared to occur frequently across different academic disciplines but were not listed in GSL. In other words, she defined common academic words as the word families that appeared frequently in academic texts but do not appear frequently in the general service domain.
A recurring theme in the question of what constitutes a good corpus, which would be used to compile a list of common vocabulary, is the notion of representativeness. Therefore, for the purposes of compiling a list of common academic vocabulary, a great deal of the efforts undertaken so far have gone towards making sure all academic disciplines are taken into consideration in building such corpora. To this end, Coxhead (2000) built Academic Corpus using selected texts from the four main disciplines of Arts, Commerce, Law, and Science, each of which included a number of subject areas. Table 2 shows some more details about the Academic Corpus.

Table 2. The structure of the Academic Corpus.

<table>
<thead>
<tr>
<th>Academic disciplines</th>
<th>Arts</th>
<th>Commerce</th>
<th>Law</th>
<th>Science</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words</td>
<td>883,214</td>
<td>879,547</td>
<td>874,723</td>
<td>875,846</td>
<td>351,333</td>
</tr>
<tr>
<td>Texts</td>
<td>122</td>
<td>107</td>
<td>72</td>
<td>113</td>
<td>414</td>
</tr>
<tr>
<td>Subject areas</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>28</td>
</tr>
</tbody>
</table>
1.3 THE TECHNICAL DOMAIN: VOCABULARY IN ESP

Efforts in compiling the Academic Word List—and other similar wordlists—were largely motivated by the idea that while academic English does use words that are generally common in English, there are also words that are central in academic writing but are not very common in the general domain, which, it was argued, necessitated the creation of a list of the most common words in academic English. In much the same way as it has been argued that academic English is different from the general domain, researchers have also argued that even within academic English, there appear to exist many language features—including vocabulary—that differ from one discipline to the other (Biber, Conrad & Reppen, 1994). The existence of different vocabulary features across all disciplines and subject areas motivated the scholars to approach the task of identifying common vocabulary from an English for Specific Purposes perspective. In other words, since a corpus intended to be used for the purposes of compiling a list of common vocabulary had to be representative of a particular register of language, scholars began to build corpora of texts in a certain subject area and to use those corpora to identify what the common vocabulary was in a particular subject area since, they argued, words behave in different ways depending on the discipline (Hyland and Tse, 2007). An example of a specialized corpus designed specifically to identify core vocabulary in a particular subject area is Crawford Camiciottoli (2007). She built the Business Studies Lecture Corpus in order to focus on the lexical items commonly used in Business English. This corpus and other similar corpora (Business English Corpus, British National Corpus, etc.) have contributed to developing Business English dictionaries as well as other language learning materials for Business major students. The Medical field, Engineering, Finance, Agriculture and Journalism are other
specialized fields in which scholars have developed specialized corpora in order to identify the
distinctive language features common to that particular discipline (Coxhead, 2012).
CHAPTER 2

METHODOLOGY

As mentioned earlier in the previous chapter, a recurring theme in the course of evolution of all corpus-based approaches to identify common vocabulary is the notion of representativeness. The Academic Word List was compiled because Coxhead (2000) believed that the General Service List collection developed earlier but West (1953) was not representative of academic English. Similarly, it was later argued that, while better than GSL, a corpora built from texts selected from a wide range of disciplines would not be very representative of each of the contributing disciplines, hence the paradigm shift of developing ESP wordlists.

The present study introduces a new approach to the task of compiling a list of common vocabulary. It is proposed that by using a number of Natural Language Processing (NLP) techniques, it is possible to write programs that can compile a list of common vocabulary using any corpus provided by the user. In other words, since the scholars’ deepest concern, throughout the years, has been to ensure the representativeness of the corpus to be used to compile a list of common vocabulary, I argue that another approach would be to allow users to bring their own data using which to compile a wordlist and that—once the burden of having a representative corpus has been outsourced to the end-user—we may focus instead on how best to compile a list of the most frequent vocabulary in the corpus. It is conceivable that if learners could use a program to generate a wordlist out of any corpus that they provide, they can essentially have a wordlist within any specialized genre or subgenre of English—or any other language for that matter—that their language needs are associated with.
2.1 DATASET AND TECHNIQUES

In order to show that using NLP techniques can yield favorable results regardless of what dataset is used, a less commonly used corpus in the field of Corpus Linguistics is used for this study. The corpus used in this study is the English-only text taken from Europarl, a corpus of “parallel text in 11 languages from the proceedings of the European Parliament, which are published on the web” (Koehn, 2005, p. 1). Europarl is among the most popular datasets used in Machine Translation because it offers parallel data in multiple languages, which can be used to build modern Statistical Machine Translation systems. The English-only part of Europarl consists of more than 2 million sentences and contains 53,974,751 words. A number of language processing tasks are required in order to compile a list of the most frequent words in a dataset. For these tasks, a few text processing libraries in Natural Language Toolkit (NLTK 3.0) were used. NLTK is a free NLP toolkit available for the programming language Python. In this study, Python 2.7.0 was used. The following section explains the algorithm used in the program in pseudocode.

2.2 PSEUDOCODE AND ALGORITHM

1 Prompt user to enter the number of most frequent nouns to find; store integer in memory.
2 Prompt user to enter the number of most frequent adjectives to find; store integer in memory.
3 Prompt user to enter the number of most frequent verbs to find; store integer in memory.
4 Import SYS and NLTK.
5 Import all NLTK libraries.
6 Open text file and store in memory.
7 Read from the opened file and store in memory.
8 Create three empty lists for verbs, nouns, and adjectives.
9 Iterate over each line of the text and for each line:
10 Tokenize the line using NLTK and store in memory.
11 Assign a Part-of-Speech tag to each tokenized line and store in memory.
12 For each ‘word and tag’ pair previously stored in memory:
13 IF the tag is ‘noun’ THEN:
14 Add the word to the list of nouns.
15 ENDIF
16 IF the tag is ‘adjective’ THEN:
17 Add the word to the list of adjectives.
18 ENDIF
19 IF the tag is ‘verb’ THEN:
20 Add the word to the list of verbs.
21 ENDIF
22 Compute the frequency distribution of the number of nouns, adjectives and verbs
that the user requested.
23 Print noun, adjective and verb frequencies.

2.3 DESCRIPTION OF THE ALGORITHM

When the program is run, the user is prompted to enter the number of the most frequent words
that the user would like the program to compute. This is done for ‘nouns’, ‘adjectives’, and
‘verbs’ and the three numbers are stored in memory for later use. Next, the program imports the
system-specific parameters and functions (SYS) and NLTK along with all its libraries. What the
program does next is use SYS to open and read the Europarl corpus. In the For-loop that comes
next, the program iterates over each line of text and breaks each line of text into its constituent words, which is a fundamental process in NLP known as tokenization. Tokenization is the process of breaking a string of letters—in this case a ‘sentence’—into smaller parts, or tokens, that are needed for the task at hand. In this case, since the intention is to perform a word-level analysis, NLTK’s built-in `word_tokenize()` function is used, which takes a string as input and returns a list of tokens where tokens are what is conventionally known as ‘words’ in English. As an example, if we were to use the built-in `word_tokenize()` function in NLTK to tokenize the sentence “This is a sentence”, the output would be `['this', 'is', 'a', 'sentence']` where `[]` are the Python notation for a list and ‘this’, ‘is’, ‘a’, and ‘sentence’ are the tokens stored in the list.

While still in the For-loop that tokenizes each line of text, the program uses the `pos_tag()` built-in function to parse each tokenized line of text and assign a part-of-speech (POS) tag to each word in the sentence. The `pos_tag()` function takes a list of tokens and returns a list of tuples where each tuple consists of a word and its assigned POS tag. For example, calling the `pos_tag()` function on the above example would print the following output:

```
[('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('sentence', 'NN'), ('.', '.')]  
```

In this output, ‘DT’, ‘VBZ’, and ‘NN’ are POS tags that belong to the Penn Treebank POS Tag Set, in which they stand for ‘determiner’, ‘third-person singular present tense verb’ and ‘singular or mass noun’, respectively. Similarly, `pos_tag()` would parse the sentence ‘Students upset about the rising cost of tuition staged a rally yesterday’ as follows:

```
[('Students', 'NNS'), ('upset', 'VBN'), ('about', 'IN'), ('the', 'DT'), ('rising', 'NN'), ('cost', 'NN'), ('of', 'IN'), ('tuition', 'NN'), ('staged', 'VBD'), ('a', 'DT'), ('rally', 'NN'), ('yesterday', 'NN'), ('.', '.')]  
```

The function would correctly identify ‘Students’ as ‘plural noun’, ‘upset’ as ‘past participle verb form’, ‘about’ as ‘preposition’, ‘the’ as a ‘determiner’, ‘cost’ as ‘noun’, ‘of’ as ‘preposition’,
‘tuition’ as ‘noun’, ‘staged’ as ‘past tense verb’, and ‘rally’ as ‘noun’. However, it incorrectly
tags “rising” and “yesterday” as ‘nouns’, whereas the correct tags would have been ‘adjective’
and ‘adverb’, respectively.
By the time this process is complete, each word in the corpus will have a POS tag assigned to it.
Knowing the POS tag of each token in the corpus provides a means of weeding out the function
words in the corpus, which would likely be the most common tokens in any corpus. Having
assigned POS tags to the tokens of each line, separating the content words from the function
words is achieved by iterating over each POS tagged line of tokenized text and adding nouns,
verbs and adjectives to the lists that were created earlier in the program. After storing all nouns,
adjectives and nouns in their corresponding lists, the program calls the FreqDist() function in
NLTK to perform frequency distribution analysis. This function can also take a number and
return the provided number of the most frequent tokens along with their counts. Using the
number that the user was prompted to enter earlier, the program can now return lists containing
the same number of the most frequent number of tokens for nouns, adjectives, and verbs along
with the frequency distribution of each token. Since these frequencies are automatically sorted
from highest to lowest, the output of the program is essentially three wordlists that show the
most frequent tokens and each token’s frequency for all nouns, adjectives, and verbs in the
corpus.
Should the user be interested in seeing examples of each of the frequent tokens, the program can
also retrieve a user-supplied number of example sentences from the original corpus where each
token was used. The following piece of pseudocode shows how sample sentences from the
corpus could be found and printed from the list of ‘nouns’. This could be modified to print
sample sentences from the list of ‘adjectives’ and ‘verbs’ as well with minor modifications.
2.4 PSEUDO CODE FOR PRINTING EXAMPLE SENTENCES

1 Prompt the user to enter a number for how many example sentence needs to be printed, save in memory.

2 Iterate over each pair of ‘word and frequency’ in the list of frequent ‘noun’:

3 Initiate a variable and set it to 0.

4 Iterate over each line of text in the corpus:

5 IF the ‘word’ in the current loop is in that line THEN:

6 Increment the value of the variable initiated above by one.

7 IF the value of the variable is less than or equal to the number that the user provided before THEN:

8 Print the line.

9 ENDIF

10 ENDIF
CHAPTER 3

RESULTS

Since running the program on the actual corpus would require computational power that is not available on standard personal computers, the program was run only on the first 100,000 lines of text in the corpus with a running word size of 2,537,838 which corresponds to about 5% of the entire corpus. Tables 3, 4, and 5 show the output of the program for the most frequent nouns, adjectives, and verbs, respectively, from the selected portion of the corpus.

Table 3: The 50 most frequent nouns and their frequencies.

<table>
<thead>
<tr>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>report</td>
<td>5446</td>
<td>16</td>
<td>part</td>
<td>1723</td>
<td>31</td>
<td>matter</td>
<td>1506</td>
<td>46</td>
<td>principle</td>
<td>1227</td>
</tr>
<tr>
<td>2</td>
<td>time</td>
<td>3570</td>
<td>17</td>
<td>view</td>
<td>1720</td>
<td>32</td>
<td>position</td>
<td>1506</td>
<td>47</td>
<td>need</td>
<td>1223</td>
</tr>
<tr>
<td>3</td>
<td>policy</td>
<td>3557</td>
<td>18</td>
<td>support</td>
<td>1712</td>
<td>33</td>
<td>process</td>
<td>1492</td>
<td>48</td>
<td>programme</td>
<td>1217</td>
</tr>
<tr>
<td>4</td>
<td>way</td>
<td>2839</td>
<td>19</td>
<td>information</td>
<td>1710</td>
<td>34</td>
<td>resolution</td>
<td>1490</td>
<td>49</td>
<td>vote</td>
<td>1210</td>
</tr>
<tr>
<td>5</td>
<td>fact</td>
<td>2739</td>
<td>20</td>
<td>case</td>
<td>1695</td>
<td>35</td>
<td>agreement</td>
<td>1438</td>
<td>50</td>
<td>law</td>
<td>1168</td>
</tr>
<tr>
<td>6</td>
<td>issue</td>
<td>2467</td>
<td>21</td>
<td>system</td>
<td>1685</td>
<td>36</td>
<td>place</td>
<td>1405</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>order</td>
<td>2330</td>
<td>22</td>
<td>year</td>
<td>1684</td>
<td>37</td>
<td>aid</td>
<td>1385</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>proposal</td>
<td>2293</td>
<td>23</td>
<td>today</td>
<td>1671</td>
<td>38</td>
<td>rapporteur</td>
<td>1346</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>development</td>
<td>2096</td>
<td>24</td>
<td>area</td>
<td>1669</td>
<td>39</td>
<td>%</td>
<td>1343</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>work</td>
<td>2082</td>
<td>25</td>
<td>problem</td>
<td>1605</td>
<td>40</td>
<td>right</td>
<td>1338</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 3 (cont.)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>point</td>
<td>2075</td>
<td>26</td>
<td>action</td>
<td>1604</td>
<td>41</td>
</tr>
<tr>
<td>12</td>
<td>situation</td>
<td>1977</td>
<td>27</td>
<td>example</td>
<td>1549</td>
<td>42</td>
</tr>
<tr>
<td>13</td>
<td>course</td>
<td>1953</td>
<td>28</td>
<td>cooperation</td>
<td>1534</td>
<td>43</td>
</tr>
<tr>
<td>14</td>
<td>market</td>
<td>1841</td>
<td>29</td>
<td>number</td>
<td>1521</td>
<td>44</td>
</tr>
<tr>
<td>15</td>
<td>country</td>
<td>1747</td>
<td>30</td>
<td>level</td>
<td>1516</td>
<td>45</td>
</tr>
</tbody>
</table>

The list of the most frequent ‘nouns’ in the corpus has the word ‘report’ as the most frequent token in the corpus with the tag ‘noun’ assigned to it by the pos_tag() function with a frequency of 5,446. In this list, the range of the frequencies of the most frequent to the least frequent is 5446 to 1,168, which belongs to the word ‘law’.

Moreover, an examination of the list of most frequent adjectives in the corpus shows the word ‘European’ to be the most frequent token with the tag ‘adjective’ in the corpus with a frequency of 8,295. The adjective ‘aware’ seems to be at the bottom of the list with a frequency count of only 701. Finally, the program identifies ‘be’ to be the most common base form of the verb in the corpus, with a frequency count of 23,657. In this list, the least frequent token is identified to be ‘look’, which appeared only 361 times.
Table 4. The 50 most frequent adjectives and their frequencies.

<table>
<thead>
<tr>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>European</td>
<td>8295</td>
<td>16</td>
<td>own</td>
<td>1606</td>
<td>31</td>
<td>few</td>
<td>1124</td>
<td>46</td>
<td>full</td>
<td>734</td>
</tr>
<tr>
<td>2</td>
<td>other</td>
<td>4061</td>
<td>17</td>
<td>able</td>
<td>1570</td>
<td>32</td>
<td>environmental</td>
<td>1117</td>
<td>47</td>
<td>large</td>
<td>731</td>
</tr>
<tr>
<td>3</td>
<td>important</td>
<td>3466</td>
<td>18</td>
<td>clear</td>
<td>1554</td>
<td>33</td>
<td>whole</td>
<td>1097</td>
<td>48</td>
<td>current</td>
<td>730</td>
</tr>
<tr>
<td>4</td>
<td>new</td>
<td>3383</td>
<td>19</td>
<td>particular</td>
<td>1544</td>
<td>34</td>
<td>next</td>
<td>1018</td>
<td>49</td>
<td>future</td>
<td>702</td>
</tr>
<tr>
<td>5</td>
<td>such</td>
<td>3047</td>
<td>20</td>
<td>last</td>
<td>1504</td>
<td>35</td>
<td>specific</td>
<td>1008</td>
<td>50</td>
<td>aware</td>
<td>701</td>
</tr>
<tr>
<td>6</td>
<td>political</td>
<td>2485</td>
<td>21</td>
<td>financial</td>
<td>1472</td>
<td>36</td>
<td>serious</td>
<td>937</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>social</td>
<td>2261</td>
<td>22</td>
<td>good</td>
<td>1463</td>
<td>37</td>
<td>general</td>
<td>907</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>many</td>
<td>2236</td>
<td>23</td>
<td>great</td>
<td>1427</td>
<td>38</td>
<td>second</td>
<td>889</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>first</td>
<td>1981</td>
<td>24</td>
<td>human</td>
<td>1406</td>
<td>39</td>
<td>various</td>
<td>845</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>same</td>
<td>1976</td>
<td>25</td>
<td>legal</td>
<td>1383</td>
<td>40</td>
<td>essential</td>
<td>774</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>possible</td>
<td>1931</td>
<td>26</td>
<td>certain</td>
<td>1382</td>
<td>41</td>
<td>real</td>
<td>769</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>economic</td>
<td>1886</td>
<td>27</td>
<td>necessary</td>
<td>1321</td>
<td>42</td>
<td>fundamental</td>
<td>767</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>directive</td>
<td>1749</td>
<td>28</td>
<td>public</td>
<td>1287</td>
<td>43</td>
<td>third</td>
<td>766</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>national</td>
<td>1711</td>
<td>29</td>
<td>international</td>
<td>1207</td>
<td>44</td>
<td>major</td>
<td>762</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>common</td>
<td>1688</td>
<td>30</td>
<td>different</td>
<td>1124</td>
<td>45</td>
<td>internal</td>
<td>739</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 5. The 50 most frequent verbs and their frequencies.

<table>
<thead>
<tr>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
<th>#</th>
<th>Token</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>be</td>
<td>23658</td>
<td>16</td>
<td>achieve</td>
<td>740</td>
<td>31</td>
<td>find</td>
<td>501</td>
<td>46</td>
<td>deal</td>
<td>372</td>
</tr>
<tr>
<td>2</td>
<td>have</td>
<td>4235</td>
<td>17</td>
<td>go</td>
<td>731</td>
<td>32</td>
<td>become</td>
<td>491</td>
<td>47</td>
<td>congratulate</td>
<td>372</td>
</tr>
<tr>
<td>3</td>
<td>like</td>
<td>4139</td>
<td>18</td>
<td>continue</td>
<td>714</td>
<td>33</td>
<td>prevent</td>
<td>481</td>
<td>48</td>
<td>consider</td>
<td>372</td>
</tr>
<tr>
<td>4</td>
<td>make</td>
<td>2807</td>
<td>19</td>
<td>help</td>
<td>631</td>
<td>34</td>
<td>bring</td>
<td>470</td>
<td>49</td>
<td>set</td>
<td>362</td>
</tr>
<tr>
<td>5</td>
<td>take</td>
<td>2753</td>
<td>20</td>
<td>create</td>
<td>615</td>
<td>35</td>
<td>adopt</td>
<td>439</td>
<td>50</td>
<td>look</td>
<td>361</td>
</tr>
<tr>
<td>6</td>
<td>say</td>
<td>1950</td>
<td>21</td>
<td>accept</td>
<td>604</td>
<td>36</td>
<td>let</td>
<td>435</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>do</td>
<td>1919</td>
<td>22</td>
<td>put</td>
<td>588</td>
<td>37</td>
<td>lead</td>
<td>420</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>see</td>
<td>1366</td>
<td>23</td>
<td>work</td>
<td>584</td>
<td>38</td>
<td>start</td>
<td>403</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>ensure</td>
<td>1323</td>
<td>24</td>
<td>get</td>
<td>563</td>
<td>39</td>
<td>think</td>
<td>385</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>give</td>
<td>1142</td>
<td>25</td>
<td>use</td>
<td>560</td>
<td>40</td>
<td>promote</td>
<td>385</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>ask</td>
<td>887</td>
<td>26</td>
<td>come</td>
<td>552</td>
<td>41</td>
<td>keep</td>
<td>381</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>support</td>
<td>855</td>
<td>27</td>
<td>improve</td>
<td>533</td>
<td>42</td>
<td>develop</td>
<td>380</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>provide</td>
<td>833</td>
<td>28</td>
<td>allow</td>
<td>523</td>
<td>43</td>
<td>protect</td>
<td>378</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>thank</td>
<td>789</td>
<td>29</td>
<td>vote</td>
<td>513</td>
<td>44</td>
<td>reduce</td>
<td>378</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>therefore</td>
<td>787</td>
<td>30</td>
<td>know</td>
<td>506</td>
<td>45</td>
<td>establish</td>
<td>378</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
As mentioned earlier, the program can also be configured such that the user may be prompted to enter a desired number of example sentences from the corpus for each identified frequent token to be printed as output (Table 5).

Table 6. Example sentences from the corpus for the tokens ‘report’, ‘competition’, and ‘aid’.

<table>
<thead>
<tr>
<th>Token</th>
<th>Example number</th>
<th>Example sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report</td>
<td>1</td>
<td>The Cunha report on multiannual guidance programmes comes before Parliament on Thursday and contains a proposal in paragraph 6 that a form of quota penalties should be introduced for countries which fail to meet their fleet reduction targets annually.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>I want to know whether one can raise an objection of that kind to what is merely a report, not a legislative proposal, and whether that is something I can competently do on Thursday.</td>
</tr>
<tr>
<td>Competition</td>
<td>1</td>
<td>Genuine structural reforms and a competition-friendly taxation policy are the cornerstones of a successful economic base.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Secondly, by adopting this directive we achieve a reduction in distortions of competition resulting from wide variations in national training structures and training costs.</td>
</tr>
</tbody>
</table>
Table 6 (cont.)

<table>
<thead>
<tr>
<th>Aid</th>
<th>1</th>
<th>As people have said, the situation there is extremely volatile.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>The relevant standards which have been laid down in another Directive, 95/35/EC, seem sufficiently adequate to advise people in a responsible manner on the organisation of the transport of dangerous goods.</td>
</tr>
</tbody>
</table>
CHAPTER 4
CONCLUSION

As mentioned earlier, the problem of representativeness seems to have been a central theme in the course of the evolution of all previous major endeavors to provide a list of common vocabulary in a language or within a certain register of a language. In the past, scholars have attempted to remedy this by first moving away from simply what is common in English to what is common in ‘academic English’ and, subsequently, identifying what is common in certain specialized subject areas within academic English. However, compiling a list of the frequent vocabulary has only been done for certain specialized field in the past, leaving many subject areas yet to be explored by corpus linguists so that similar wordlists may be generated.

In this study, it was proposed that an alternative approach would be to enable users to provide their own language input, using which a list of frequent vocabulary could be compiled. It was argued that this alternative approach would maximize representativeness in areas where wordlists have been offered in the past while enabling learners to generate wordlists in specialized areas where no wordlists have been generated in the past. Assuming that the language input—which can be any textbook or other large body of text—that the users already have would be most representative of what genre of language their needs are associated with, NLP techniques can help to compile a list of common vocabulary that directly represents the users’ target language register. The focus, then, can be placed on how to process the user-provided text to compile a list of common vocabulary. In this study, it was shown that it is possible to use a number of NLP techniques available in NLTK to generate a list of most frequent words in a corpus that the user provides as input. It was shown that this corpus could be tokenized into the words that make up the sentence. It would then be possible to iterate over the
resulting tokens to measure the frequency distribution of each word in the text. The results could then be sorted to produce a list of the most common tokens in the text. Since it is expected that most of the frequent tokens will be words that might not be technical words, a POS-tag labeling phase was introduced to filter out function words. Once a list of the most frequent nouns, verbs, or adjectives of a corpus has been compiled, it would then be possible to print a selection of sentences from the text in which each of the frequent words appeared in the original text. The output of the program would be an input-specific list of frequent vocabulary, along with example sentences that would show the words used in their original context.

4.1 PEDAGOGICAL IMPLICATIONS
Since one of the ways that identifying common vocabulary has served learners of English has been to help them set vocabulary-learning goals (Coxhead, 2000), it is arguable that the present approach could be more beneficial for learners since it would give them the opportunity to have a contextualized list of the frequent words, as opposed to a list of common words out of context. The present methodology would also be helpful because the example sentences generated are taken from the actual corpus that the user provides rather than a generic sentence, which means that the example sentences would also be authentic. In other words, the present method adds contextualization and authenticity—both of which are important characteristics of valuable learning materials—to the luxury of having a list of common vocabulary that is specific to a user-provided corpus as opposed to a generic wordlist. Other than contextualization and authenticity, the present approach would also benefit learners by helping them have wordlists in specialized areas that there might not already exist any (recent) wordlists. It was mentioned earlier that within the area of English for Specific Purposes, learners now have access to
wordlists in such areas as Law, Engineering, Business, and so on. Using the mechanism offered in this study, however, learners can generate wordlists in any area in which a wordlist might not already exist.

Another major advantage of using NLP techniques for the purposes of identifying frequent words is that these techniques are not language specific and thus can be applied to other languages as well. Using the procedure explained in the present study, for example, it would be possible to generate a list of common words in other languages as well if a POS tagger is available for those languages. For instance, the Stanford Parser offers free POS taggers for German, French, Chinese, and Arabic, meaning it would be possible to compile a list of the most frequent tokens from any text in these languages following the same algorithm as proposed in this study. NLTK’s FreqDist() function could still be used to compute and sort the word counts even if another POS tagger were to be used instead of NLTK’s pos_tag() tagger; however, it would also be possible to write functions that process the data and keep counts of all tokens and then sort and print the tokens with the highest counts which would correspond to the most frequent tokens in the corpus as computed by NLTK’s built-in tagger. For languages for which an automated POS-tagger is not already available, however, either another mechanism for separating content and function words would need to be used, or a POS tagger would need to be trained.
4.2 LIMITATIONS AND SUGGESTIONS FOR FUTURE RESEARCH

A crucial step in identifying the most frequent words in the present approach is parsing each sentence and assigning POS tags to each word. It should be noted, however, that automated POS-tagging does not yield 100% accuracy and could potentially lead to inaccurate results. For instance, as mentioned earlier, NLTK parses the sentence ‘Students upset about the rising cost of tuition staged a rally yesterday’ as follows:

[('Students', 'NNS'), ('upset', 'VBN'), ('about', 'IN'), ('the', 'DT'), ('rising', 'NN'), ('cost', 'NN'), ('of', 'IN'), ('tuition', 'NN'), ('staged', 'VBD'), ('a', 'DT'), ('rally', 'NN'), ('yesterday', 'NN'), ('.', '.')]

In this example, “rising” and “yesterday” are incorrectly tagged as ‘nouns’, whereas the correct tags would have been ‘adjective’ and ‘adverb’, respectively. Upon examination of the results generated by the program (Tables 2, 3, and 4), however, it appears that the most frequent tags are in the right table; that is, all of the frequent tokens identified as frequent nouns are nouns indeed as is the case with the adjectives and verbs. In other words, despite the fact that the process of automated POS tagging is not 100% accurate (to determine the actual accuracy, one would have to manually annotate a corpus, compare the annotations with those created by NLTK and then compute the percentage of times NLTK was correct), it still offers a reliable means of separating the most frequent nouns, verbs, and adjectives in a large corpus. This might be due to the fact that even if some words were to be incorrectly tagged, they would still be outnumbered by correctly-tagged words and since in the following stage all words and their tags would be counted, the most frequent tokens would still be the ones that bear the right tags.

In NLP, automated POS-tagging is achieved by first having a training corpus which has been previously tagged by human annotators for part-of-speech. Using this training set, automated POS-taggers learn the distribution of the probabilities of each POS tag that any given word has in
the training (seen) corpus and later use these conditional probabilities to compute the most likely set of tags for any unseen set of words. The accuracy of applying this procedure to unseen data would then depend on how much training data the parser has seen before, the accuracy of the original human-annotated corpus, and most importantly, whether the training set and the test set are similar in terms of linguistic features. For the purposes of this program, one source of inaccuracy in the performance of NLTK’s `pos_tag()` function might be that the function is being used on language input that might not share the same linguistic features as the training data that NLTK’s `pos_tag()` has been trained on. As mentioned earlier, however, and given the results of the program, it seems that using automated POS-tagging does offer a reliable means of separating content and function words and, more importantly, of generating a list of common words in a corpus that all share a certain part of speech (i.e., a list of ‘nouns’, ‘verbs’, ‘adjectives’ and so on).

Looking at the example sentences that the program generated from the corpus for each word, another area for improvement in this work is adding a stage of tokenization before example sentences are printed. More specifically, in the current implementation of the program any sentence from the text that contains the string of letters corresponding to the ones in a particular frequent word anywhere in the sentence gets identified as an example sentence for that word. For instance, as shown in Table 5, one of the example sentences that the program prints for the word ‘aid’ is the following:

*The relevant standards which have been laid down in another Directive, 95/35/EC, seem sufficiently adequate to advise people in a responsible manner on the organisation of the transport of dangerous goods.*
Although the word “aid” does not appear in the sentence, this sentence is identified as an example of “aid” because the word “laid” in the sentence has the string of letters aid in it and hence matches the algorithm’s string-matching way of identifying example sentences. To remedy this, the algorithm could be modified such that the program looks for a certain string (i.e., frequent word) in a tokenized sentence. If this were to be implemented, “laid” would no longer be a match for a string-matching search for “aid” and therefore the above error could be avoided.

Another potential area of inaccuracy in the performance of this program might be how the most frequent verbs are identified and counted. For the sake of simplicity, in the current implementation of the program only the base forms of verbs are taken into consideration. An improved approach would be to include all verb forms and all tenses into consideration, in which case there would also have to be a stemming process that reduces each verb form to its base form in order to increment any verb form’s base form count whenever any of the verb’s various forms appear in the text.

Additionally, it should be acknowledged that the present implementation of the program is only capable of performing word-level analysis and completely ignores any phrasal expressions that might be frequent in the corpus as well as all information about collocations in the text. Since collocation information is an important aspect of vocabulary learning, another area for improvement for the program would be to include N-gram language modeling in the program such that frequent two-word, three-word, or N-word expressions in the text could also be captured.

It is expected that this approach could be useful as a place to start to prepare instructional materials for any given academic discipline, given that a representative textbook from that discipline is available in digital format that could be used as a working corpus.
REFERENCES


APPENDIX A: PROGRAM SCRIPT FOR PYTHON 2.7.0

nouns_num = int(input('How many of the most frequent nouns would you like to see? '))
adjectives_num = int(input('How many of the most frequent adjectives would you like to see? '))
verbs_num = int(input('How many of the most frequent verbs would you like to see? '))

import sys, nltk
from nltk import *
op = open(sys.argv[1])
lines = op.readlines()
verbs = []
nouns = []
adjectives = []

for line in lines:
    tokenized_line = word_tokenize(line)
tagged = pos_tag(tokenized_line)
for (word, tag) in tagged:
    if tag == 'NN':
        nouns.append(word)
    elif tag == 'JJ':
        adjectives.append(word)
    elif tag == 'VB':
        verbs.append(word)

noun_freqs = FreqDist(nouns).most_common(nouns_num)
verb_freqs = FreqDist(verbs).most_common(verbs_num)
adjective_freqs = FreqDist(adjectives).most_common(adjectives_num)
print(noun_freqs, '\n')
print(verb_freqs, '\n')
print(adjective_freqs, '\n')

N = int(input('How many example sentences would you like to see for each word? '))
for (word, _) in noun_freqs + verb_freqs + adjective_freqs:
    num = 0
    for line in lines:
        if word in line:
            num += 1
        if num <= N:
            print('Example number {} for "{}":'.format(num, word))
            print(line, '\n')