```python
# Appendix C5 - feature_extractor.py

from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import nltk

class FeatureExtractor():
    def __init__(self, documents, n_features=5000):
        self.documents = documents
        self.stemmer = PorterStemmer()
        self.vocabulary = self.top_words(n_features, self.freq_dist(self.make_vocabulary(
)))

    def tokenize(self, document=None):
        if document:
            documents = [document]
        else:
            documents = self.documents

        return [token for doc in documents for token in word_tokenize(doc.content)]

    def process(self, vocabulary):
        ADDITIONAL_STOP_WORDS = {'january', 'please', 'https', 'email', 'detail', 'email'
, 'send', 'if', 'december', 'october', 'kb', 'february', 'within', 'november', 'may', 'plea
se', '.mb', 'what', 'pdf', 'june', 'mach', 'good', 'august', 'september', 'html', 'july', '
beta', 'document', 'eg', 'published', 'april'}
        stop_words = set(stopwords.words("english"))

        processed_words = []
        for word in vocabulary:
            # select only words shorter than 20 char
            if len(word) < 20:
                word = word.lower()
                # do not select stopwords
                if word not in (stop_words | ADDITIONAL_STOP_WORDS):
                    # stem words
                    word = self.stemmer.stem(word)
                    # do not select words shorter than 2 characters
                    if word.isalpha:
                        if len(word) > 1:
                            processed_words.append(word)
                    else:
                        processed_words.append(word)
        return processed_words

    def make_vocabulary(self, document=None):
        if document:
            vocabulary = self.tokenize(document)
        else:
            vocabulary = self.tokenize()

        vocabulary = self.process(vocabulary)
        return vocabulary

    def bag_of_words(self, document):
        doc_words = set(self.make_vocabulary(document))
        bag_of_words = {}

        for word in self.vocabulary:
            bag_of_words[word] = (word in doc_words)

        return bag_of_words

    def freq_dist(self, vocabulary):
        return nltk.FreqDist(vocabulary)

```

```
64      def top_words(self, n_features, freq_dist):
65          return list(freq_dist.keys())[:n_features]
66
```