1.Duży (> 1000 elementów) zbiór dokumentów tekstowych w języku angielskim

```
def generete_texts():
    for i in range(1001):
        gen = MarkovTextGenerator()
        file_name = 'text' + str(i) + '.txt'
        path = 'texts/' + file_name
        string_to_write = gen.gen_text(100)
        file = open(path, 'w')
        file.write(string_to_write)
        file.close()
        # os.remove(path)
```

2.Słownik słów kluczowych (termów)- vocab w funkcji generate_bow()

3. Wektor cech bag-of-words bag_vector[i] w funkcji generate_bow()

4. Macierz wektorów cech term-by-document matrix – transpose_list w funkcji generate_bow()

```
def word_extraction(sentence):
    ignore = ['a', "the", "is"]
    words = re.sub("[^\w]", " ", sentence).split()
    cleaned_text = [w.lower() for w in words if w not in ignore]
    cleaned_text2 = [w.replace(",", "") for w in cleaned_text]
    cleaned_text3 = [w.replace(".", "") for w in cleaned_text2]
    return cleaned_text3
```

```
def tokenize(sentences):
    words = []
    for sentence in sentences:
        w = word_extraction(sentence)
        words.extend(w)
        words = sorted(list(set(words)))
    return words
```

```
def texts_to_array():
    allsentences = []
    for i in range(1,5):                                #zmienić na 1001
        file_name = 'text' + str(i) + '.txt'
        path = 'texts/' + file_name
        sentence = Path(path).read_text()
        allsentences.append(sentence)
    return allsentences
```

```
def generate_bow():
    allsentences = texts_to_array()
    global vocab
    vocab = tokenize(allsentences)
    # print("Word List for Document \n{0} \n".format(vocab))
    bag_arr = []
    LEN_VOC = len(vocab)
    for sentence in allsentences:
        words = word_extraction(sentence)
        bag_vector = [0]*LEN_VOC
        for w in words:
            for i,word in enumerate(vocab):
                if word == w:
                    bag_vector[i] += 1

        bag_arr.append(bag_vector)


        #print("{0}\n{1}\n".format(words,numpy.array(bag_vector)))
    numpy_array = numpy.array(bag_arr)
    transpose = numpy_array.T
    global matrix_before_trans
    matrix_before_trans = numpy_array.tolist()
    transpose_list = transpose.tolist()
    return transpose_list, vocab
```

5.Przetworzony wstępnie otrzymany zbiór(document_matrix) danych mnożąc elementy bag-of-words przez inverse document frequency

```
def inverse_document_frequency():
    global document_matrix
    document_matrix, voc = generate_bow()
    LEN_VOC = len(vocab)
    counter_words = [0] * (LEN_VOC)
    for i in range(LEN_VOC):
        for j in range(4):                  #zmienić na 1001
            if document_matrix[i][j] != 0:
                counter_words[i] += 1
        if counter_words[i] != 0:
            counter_words[i] = math.log(1001/counter_words[i])

    for i in range(4):                  #zmienić na 1001
        for j in range(LEN_VOC):
            document_matrix[j][i] *= counter_words[j]

    return document_matrix
```

6. Zapytanie k najbardziej podobnych dokumentów(nie działa)

```
def input_to_bag_of_words():
    input_string= input(">>")
    LEN_VOC = len(vocab)
    words = word_extraction(input_string)
    bag_vector = [0] * LEN_VOC

    for w in words:
        for i, word in enumerate(vocab):
            if word == w:
                bag_vector[i] += 1

    return bag_vector

def cos(vector1, vector2):
    cos = 0
    for i in range(len(vector1)):
        cos += (vector1[i]*vector2[i])
    len1 = 0
    len2 = 0
    for i in range(len(vector2)):
        len1 += vector1**2
        len2 += vector2**2
    len1 = numpy.sqrt(len1)
    len2 = numpy.sqrt(len2)

    cos = cos/(len1*len2)
    return cos

def k_most_similiar_texts(k):
    compare_document = []
    print((matrix_before_trans))
    for i in range(1001):
        familiar = cos(matrix_before_trans[i],input_to_bag_of_words())
        compare_document.append(familiar, i)
    compare_document.sort(key = lambda x: x[0])

    print("the most familiar texts:")

    for i in range(k):
        file_name = 'text' + str(compare_document[i][1]) + '.txt'
        path = 'texts/' + file_name
        print("{0} similiarity: {1}".format(path, compare_document[i][0]))
```