1 Problem with solution 20

1.1 TSNE representation

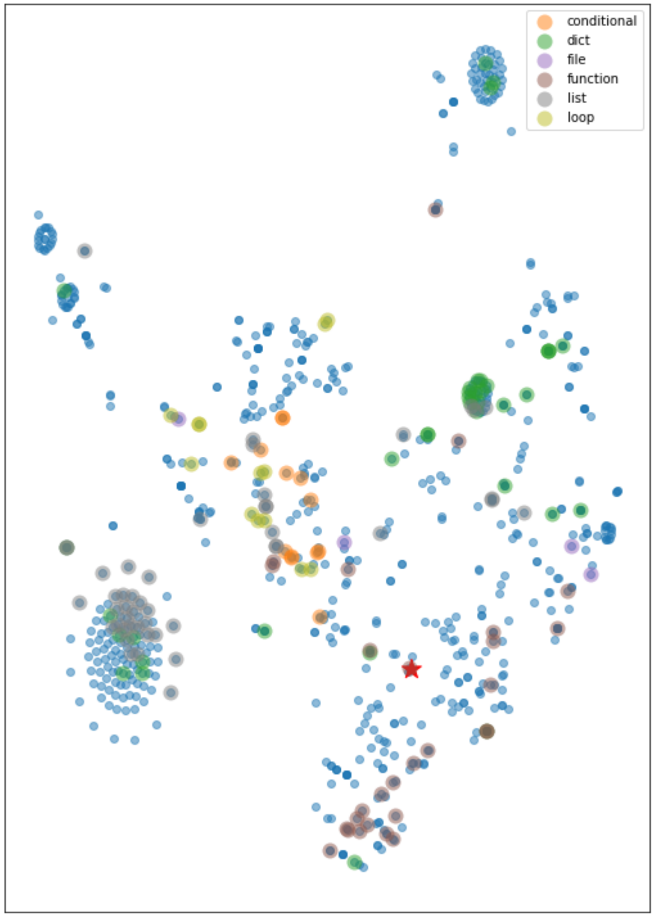

Figure 1: TSNE 2d feature space

1.2 Problem and Solution

Ask the user for a number and determine whether the number is prime or not. (For those who have forgotten, a prime number is a number that has no divisors.). You can (and should!) use your answer to Exercise 4 to help you. Take this opportunity to practice using functions, described below.

```
def get_number(prompt):
    '''Returns integer value for input. Prompt is displayed text'''
    return int(input(prompt))


def is_prime(number):
    '''Returns True for prime numbers, False otherwise'''
    #Edge Cases
    if number == 1:
        prime = False
    elif number == 2:
        prime = True
    #All other primes
    else:
        prime = True
        for check_number in range(2, (number / 2)+1):
            if number % check_number == 0:
                prime = False
                break
    return prime

def print_prime(number):
    prime = is_prime(number)
    if prime:
        descriptor = ""
    else:
        descriptor = "not "
    print(number," is ", descriptor, "prime.", sep = "", end = "\n\n")



#never ending loop

while 1 == 1:
    print_prime(get_number("Enter a number to check. Ctl-C to exit."))
```

2 Problem with solution 21

2.1 TSNE representation

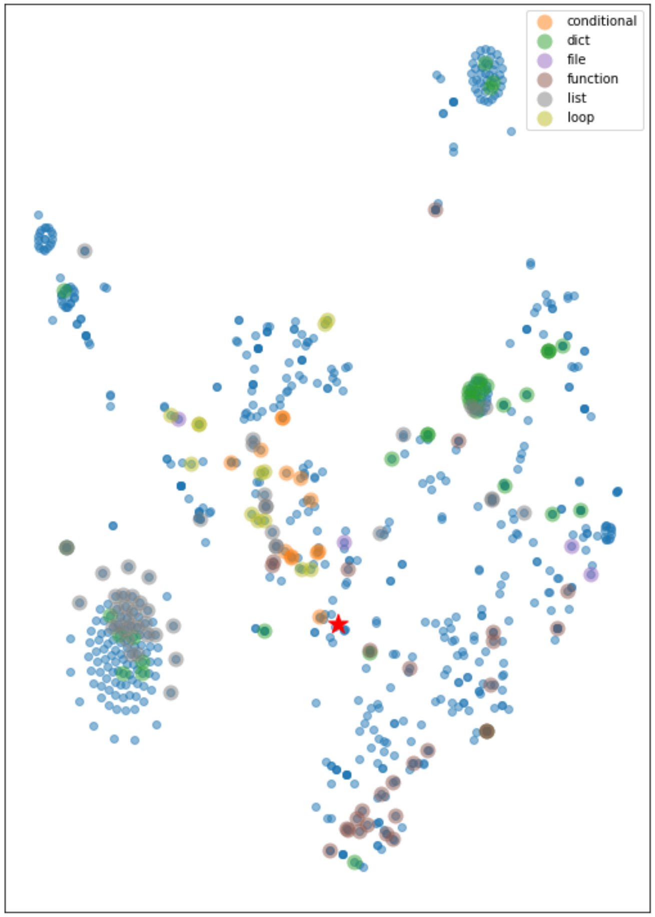

Figure 2: TSNE 2d feature space

2.2 Problem and Solution

Ask the user for a number and determine whether the number is prime or not. (For those who have forgotten, a prime number is a number that has no divisors.). You can (and should!) use your answer to Exercise 4 to help you. Take this opportunity to practice using functions, described below.

```
import sys
number = input("Please enter a number" + "\n" + ">>>")
number = int(number)
prime = False #initiate boolean for true false, default false
if number > 0:
    for x in range (2, number - 1): #this range excludes number and 1, both of whic
        if number % x != 0: #If number isn’t evenly divisible by x, start over with
            continue
        elif number % x == 0: #If number is evenly divisible by x, it can’t be prim
            sys.exit("The number is not prime.")
    sys.exit("The number is prime.") #number wasn’t evenly divisible by any x, so it
elif number == 0:
    sys.exit("The number is not prime.") #According to the Google, 0 is not prime
else:#if number is less than 0, the number is not prime (according to the Google).
    sys.exit("The number is not prime.")
```

3 Problem with solution 22

3.1 TSNE representation

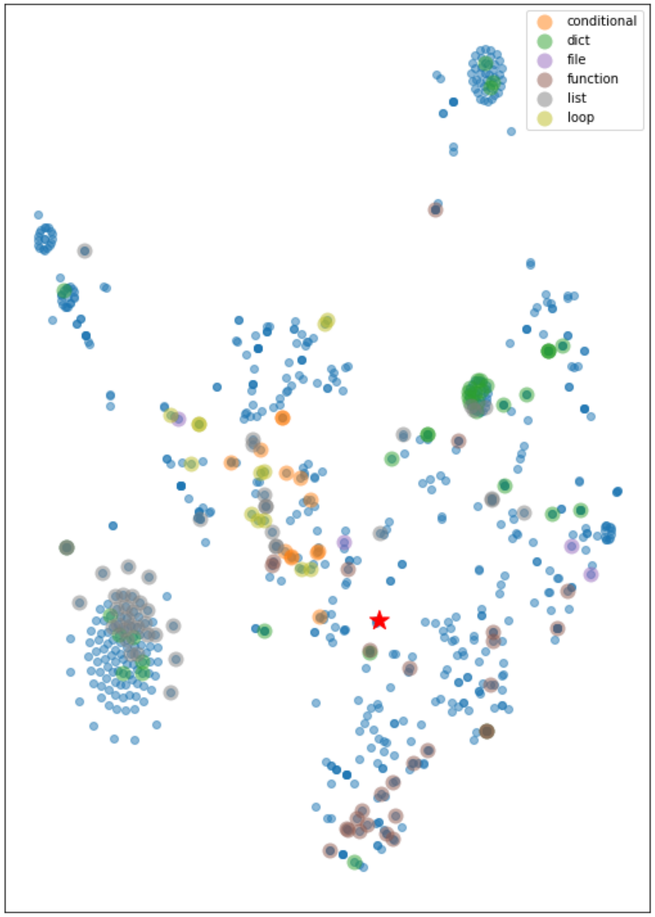

Figure 3: TSNE 2d feature space

3.2 Problem and Solution

Ask the user for a number and determine whether the number is prime or not. (For those who have forgotten, a prime number is a number that has no divisors.). You can (and should!) use your answer to Exercise 4 to help you. Take this opportunity to practice using functions, described below.

```
num = int(raw_input('Insert a number: '))
a = [x for x in range(2, num) if num % x == 0]

def is_prime(n):
if num > 1:
if len(a) == 0:
print 'prime'
else:
print 'NOT prime'
else:
print 'NOT prime'

is_prime(num)
```

4 Problem with solution 23

4.1 TSNE representation

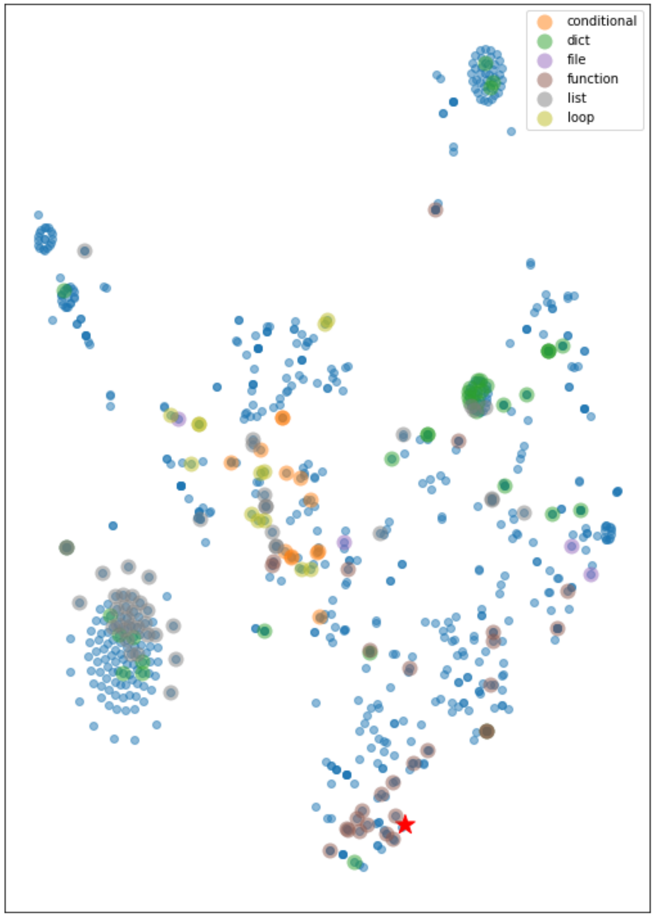

Figure 4: TSNE 2d feature space

4.2 Problem and Solution

Write a program that takes a list of numbers (for example, a = [5, 10, 15, 20, 25]) and makes a new list of only the first and last elements of the given list. For practice, write this code inside a function.

```
def list_ends(a_list):
    return [a_list[0], a_list[len(a_list)-1]]
```

5 Problem with solution 24

5.1 TSNE representation

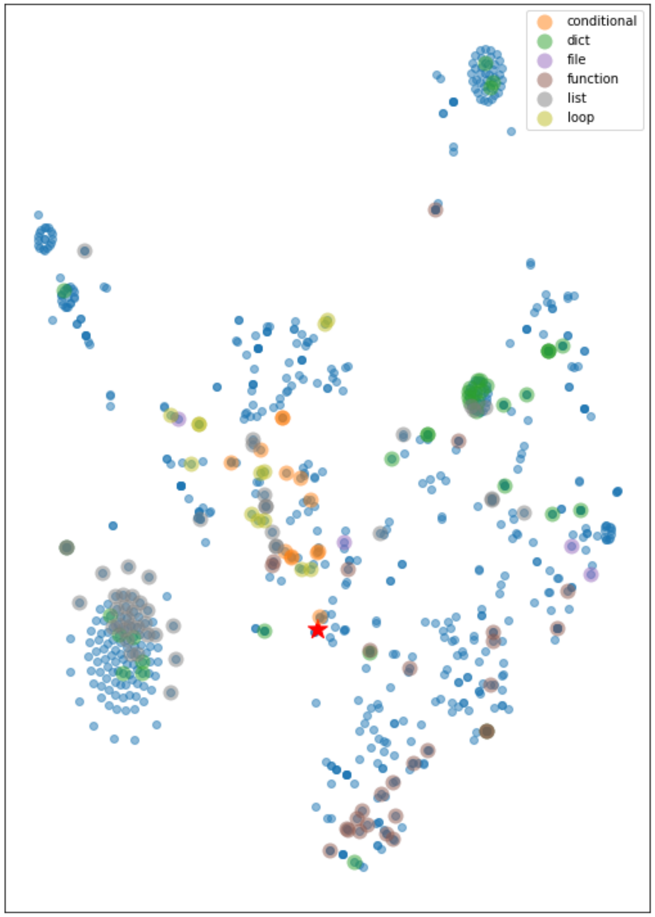

Figure 5: TSNE 2d feature space

5.2 Problem and Solution

Write a program that asks the user how many Fibonnaci numbers to generate and then generates them. Take this opportunity to think about how you can use functions. Make sure to ask the user to enter the number of numbers in the sequence to generate.(Hint: The Fibonnaci seqence is a sequence of numbers where the next number in the sequence is the sum of the previous two numbers in the sequence. The sequence looks like this: 1, 1, 2, 3, 5, 8, 13, . . .)

```
def fibonacci():
    num = int(input("How many numbers that generates?:"))
    i = 1
    if num == 0:
        fib = []
    elif num == 1:
        fib = [1]
    elif num == 2:
        fib = [1,1]
    elif num > 2:
        fib = [1,1]
        while i < (num - 1):
            fib.append(fib[i] + fib[i-1])
            i += 1
    return fib
print(fibonacci())
input()
```

6 Problem with solution 25

6.1 TSNE representation

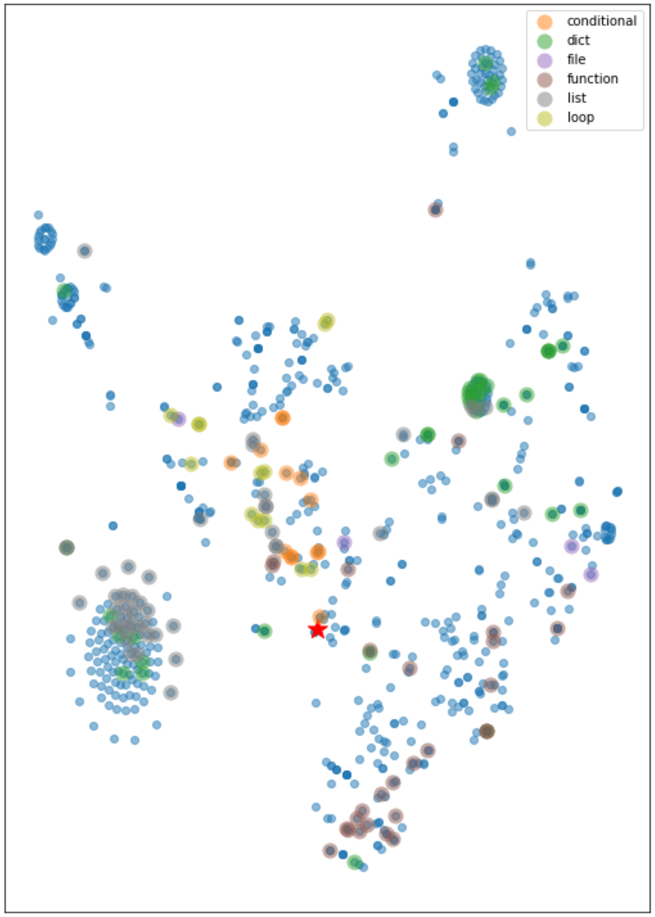

Figure 6: TSNE 2d feature space

6.2 Problem and Solution

Write a program that asks the user how many Fibonnaci numbers to generate and then generates them. Take this opportunity to think about how you can use functions. Make sure to ask the user to enter the number of numbers in the sequence to generate.(Hint: The Fibonnaci seqence is a sequence of numbers where the next number in the sequence is the sum of the previous two numbers in the sequence. The sequence looks like this: 1, 1, 2, 3, 5, 8, 13, . . .)

```
def gen_fib():
    count = int(input("How many fibonacci numbers would you like to generate? "))
    i = 1
    if count == 0:
        fib = []
    elif count == 1:
        fib = [1]
    elif count == 2:
        fib = [1,1]
    elif count > 2:
        fib = [1,1]
        while i < (count - 1):
            fib.append(fib[i] + fib[i-1])
            i += 1

    return fib
```

7 Problem with solution 26

7.1 TSNE representation

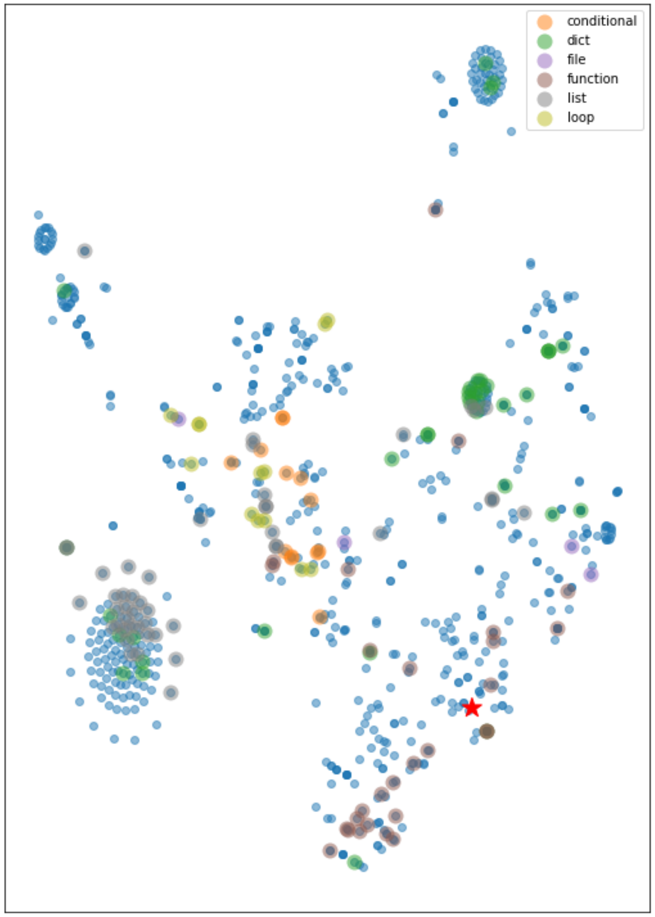

Figure 7: TSNE 2d feature space

7.2 Problem and Solution

Write a program (function!) that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates.
Extras:

Write two different functions to do this - one using a loop and constructing a list, and another using sets.
Go back and do Exercise 5 using sets, and write the solution for that in a different function.

```
# Exercise 13:
# Write a function that takes a list and returns a new list that contains
# all the elements of the first list minus duplicates.

# this one uses a for loop
def dedupe_v1(x):
  y = []
  for i in x:
    if i not in y:
      y.append(i)
  return y

#this one uses sets
def dedupe_v2(x):
    return list(set(x))

a = [1,2,3,4,3,2,1]
print a
print dedupe_v1(a)
print dedupe_v2(a)
```

8 Problem with solution 27

8.1 TSNE representation

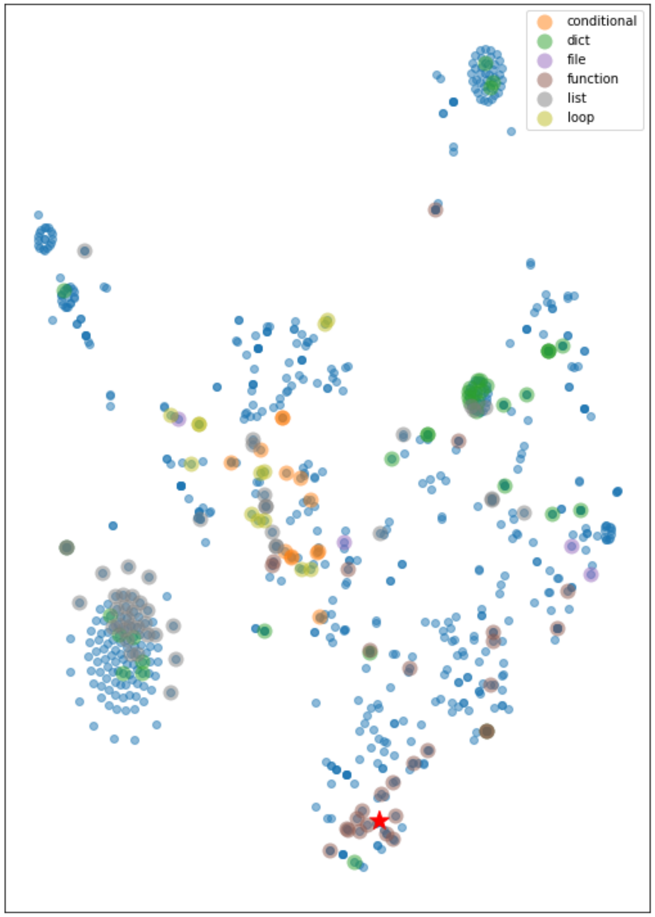

Figure 8: TSNE 2d feature space

8.2 Problem and Solution

Write a program (using functions!) that asks the user for a long string containing multiple words. Print back to the user the same string, except with the words in backwards order. For example, say I type the string:
My name is Michele
Then I would see the string:
Michele is name My
shown back to me.

```
# Exercise 14
# Write a function that asks the user for a string containing multiple words. Print
# back to the user the same string, except with the words in backwards order.

# method 1: loop through the words and insert each word at the begining of the result list
def reverse_v1(x):
  y = x.split()
  result = []
  for word in y:
    result.insert(0,word)
  return " ".join(result)

# method 2
def reverse_v2(x):
  y = x.split()
  return " ".join(y[::-1])

# method 3
def reverse_v3(x):
  y = x.split()
  return " ".join(reversed(y))

# method 4
def reverse_v4(x):
  y = x.split()
  y.reverse()
  return " ".join(y)

# test code
test1 = raw_input("Enter a sentence: ")
print reverse_v1(test1)
print reverse_v2(test1)
print reverse_v3(test1)
print reverse_v4(test1)
```

9 Problem with solution 28

9.1 TSNE representation

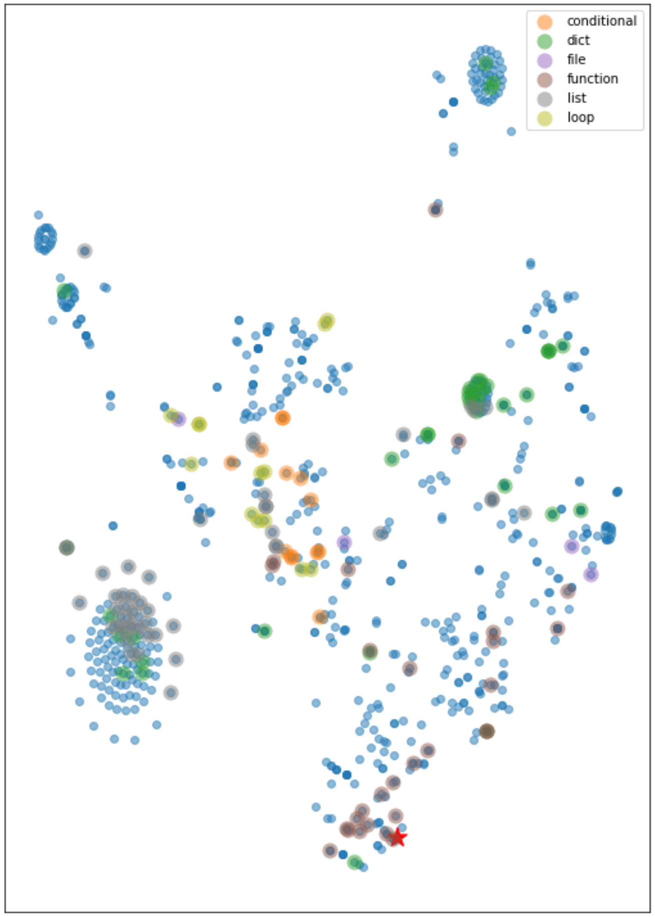

Figure 9: TSNE 2d feature space

9.2 Problem and Solution

Write a program (using functions!) that asks the user for a long string containing multiple words. Print back to the user the same string, except with the words in backwards order. For example, say I type the string:
My name is Michele
Then I would see the string:
Michele is name My
shown back to me.

```
def reverseWord(w):
  return ' '.join(w.split()[::-1])
```

10 Problem with solution 29

10.1 TSNE representation

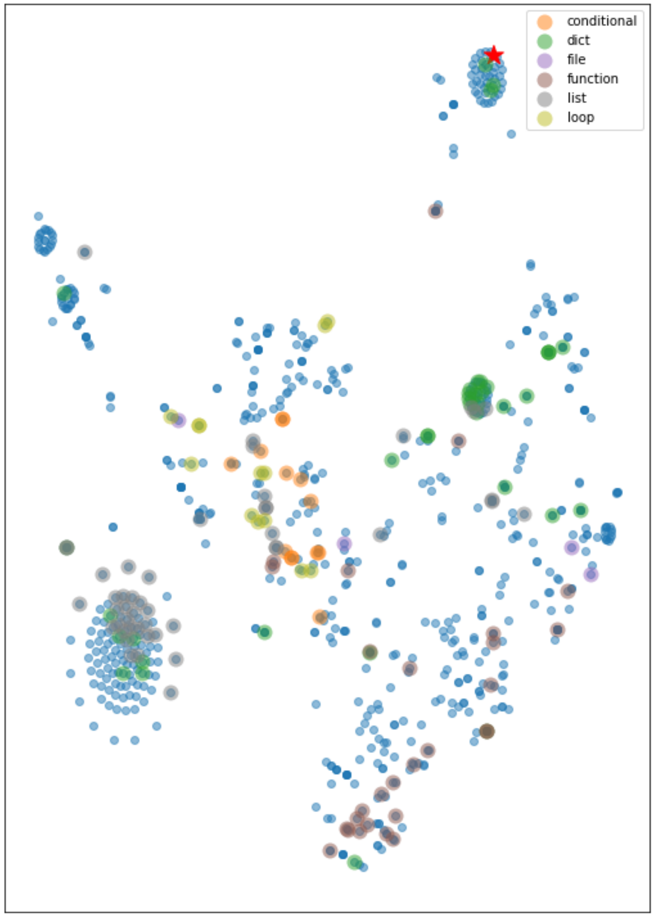

Figure 10: TSNE 2d feature space

10.2 Problem and Solution

Write a password generator in Python. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password. Include your run-time code in a main method.
Extra:

Ask the user how strong they want their password to be. For weak passwords, pick a word or two from a list.

```
# generate a password with length "passlen" with no duplicate characters in the pass

import random

s = "abcdefghijklmnopqrstuvwxyz01234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ!@#$%^&*()?"
passlen = 8
p =  "".join(random.sample(s,passlen ))
print p
```