Memoria P1

Leah Hadeed & Lorenzo Vela

1) Para el ejercicio 2, era necesario crear un buscador utilizando el modelo vectorial. Para esto, utilizamos el modelo estudiado en clase tf-idf.

```
def tf(freq):
return 1 + math.log2(freq) if freq > 0 else 0

def idf(df, n):
return math.log2((n + 1) / (df + 0.5))
```

Separamos la busqueda en dos funciones, search y score, donde score calcula el df (df = tf*idf).

```
return tf(self.index.term_freq(term, doc)) * idf(self.index.doc_freq(term), self.index.ndocs())
```

Para el VSMCosineSearcher, simplemente sobreescribimos la funcion de score y lo reeplazamos con tf*idf/ sqrt(df^2). Para calcular el modulo, creamos un fichero de referencia de modulos para luego simplemente tener que buscarlo por su id despues de calcularlo. Este fichero lo inicializamos antes de la primera consulta al VSMCosineSearcher.

```
return ( tf(self.index.term_freq(term, doc)) * idf(self.index.doc_freq(term),
self.index.ndocs()) ) / get_mod(self.index, doc)
```

2) Para conseguir que un documento se ponga al principio de la lista, es necesario que la proporcion de los terminos de la query sea alta. Por eso, incluimos un documento a mano con las palabras de la query ‘obama family tree’ repetidas.

3) Creamos el modulo statistics.py, pero no conseguimos lanzarlo porque no pudimos instalar matplotlib, entonces las lineas del codigo donde se crearian los plots estan comentados.