UNIVERSITÀ DI PISA

Fourth hands-on: Tweets

Algorithm Design (2021/2022)

Gabriele Pappalardo
Email: g.pappalardo4@studenti.unipi.it
Department of Computer Science

March 2022

1 Introduction

Given a stream of Tweets collected using Twitter, answer to these questions:

- Count the percentage of happy users in the different moments of the day (morning, afternoon, evening, night). Discuss what you find if compute also the percentage of unhappy users. Do the two percentages sum to 100%? Why?
- Spell the 30 favorite words of happy users.
- Find the number of distinct words used by happy users. How could we exclude words repeated only once?
- Decide if, in general, happy messages are longer or shorter than unhappy messages.

2 Solution

The solution for the first point uses one linear counter for each moment of the day. Thus, we are going to have a total of four counters for the happy users, and other four for the unhappy ones. If we sum the obtained percentages we could obtain a 100% under certain conditions, that is, a user should post only a tweet. Otherwise, we could never obtain the 100% since the mood of a user could change during the day (see the user *@missbrandii*, he/she has three negative tweets and a positive one inside the `sample.csv` dataset.).

In order to spell the 30 favorite words of happy users we can use the *space saving algorithm*, seen during the lecutres, and query the table at the end for the first 30 words.

To find the distinct words used by happy users, we thought to use a LogLog counter combined with a Bloom Filter. The LogLog counter is suitable to count distinct elements. Thus, to understand if a word is repeated only once we use a Bloom Filter in the following way:

1. Set the bit of the hashed value to 1;
2. If the bit was set to 1, then it means we already saw that word and therefore we count it using the LogLog counter.

A pseudo Python code solution is shown Listing 1.

```
def count_distinct_words(s: Stream) → Int:

    counter = HyperLogLogCounter( ... )
    filter = BloomFilter( ... )

    word = s.next()

    while word is not None:
        if not filter.is_set(word):
            filter.set(word)
        else:
            counter.add(word)
        word = s.next()

    return counter.count()
```

Listing 1: "Pseudo-Code for the third point"

The fourth point can be solved using several approaches, we used the *median* as indicator. In the `mini.csv` and `sample.csv` datasets, the negatives messages are longer than the positives ones. Instead, using the *mean* as indicator, in the `mini.csv` dataset, positive messages are longer.