Gorilla–Sea Cucumber Hash

Thore Husfeldt

Revision 6e6b2ae , Thu Feb 15 13:55:54 2018 +0100

Description

This is a simple data mining application that uses hashing to compare protein sequences in order to learn if we are closer (genetically) to a gorilla or a sea cucumber.

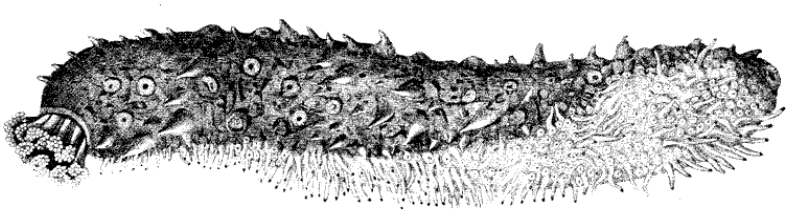

Figure 1: A sea cucumber. Ugly bastard. Image from *Nordisk familjebok*, 1876.

Implement the similarity algorithm described below. You can use whatever hash function you want, and play around with parameters k and d until you get output that you are biologically comfortable with. For me, $k = 20$ and $d = 10000$ works pretty well. You may need to consult some kind of reference work to remember (or learn) what a dot product is, and how to compute $|p|$, the length of vector p.

Input

The input file `HbB_FASTAs.in` contains proper data from some protein sequence database. Below are the first few lines of `HbB_FASTAs.in`.

```
>Human 2144721 HBHU 4HHB
MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLG
AFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVAN
ALAHKYH
>Gorilla 232230
MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLG
AFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFKLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVAN
ALAHKYH
...
```

The protein sequence for Human takes up three lines, from `MVHL` to `KYH`. It contains a list of amino acids, coded as alphabetic letters. Then, after the next `>` symbol, the file continues with the next species, Gorilla. Ignore things like `2144721 HBHU 4HHB` – I have no idea what that stuff even means!

Output

For each pair of organisms, output a number between 0 and 1 that describes their similarity. If you did everything right, organisms like Human and Gorilla should be judged more similar than Cow and Sea cucumber, which is consistent with our knowledge of evolution.

Similarity test based on hashing

For integer k, a *k-gram* of a string is a substring of length k. Clearly, every string of length n has at most $n - k$ such k-grams, but it may have fewer because of repetitions. For instance, the 2-grams of ABABAA are AB, BA, and AA. Both AB and BA appear twice.

For integer d, let h be a hash function that maps the set of k-grams to the integers $\{0, \ldots, d-1\}$. For example, in Java, you could define $h(S)$ as `(S.hashcode() % d)` . Note that we provide a python version of this hash-function that you can import with `from algs4.fundamentals.java_helper import java_string_hash`. Alternatively, you can try something like `pyfasthash`[1]. Given a string S, define for each i with $0 \leq i < d$ the value p_i as the number of k-grams T of S for which $h(T) = i$. The resulting d-dimensional vector $p = (p_0, \ldots, p_{d-1})$ is called the *profile* of S. We then define the *similarity* of two strings with profiles p and q as

$$\frac{p \cdot q}{|p|\,|q|},$$

where $p \cdot q$ is the dot product[2] of p and q, and $|p|$ is the (Euclidean) length of p.

[1] `https://github.com/flier/pyfasthash` and the installation instruction there.

[2] The dot product is sometimes called the scalar product or inner product.

Why is this this a reasonable definition of similarity? First, two identical strings have the same profile. More interestingly, two similar strings will have similar profiles: ABACADABRA and BABABLACKSHEEP contain AB the same number of times, both of which contribute to the profile index counting $h(\text{AB})$. Assuming uniform hashing, no other 2-grams are likely to get the same hash value, so the profile index is roughly the same. We can then define two profile vectors as similar if they "point in the same direction," which means that the angle between them is small. Recall that the angle α between two vectors p and q is defined as $\cos\alpha = (p \cdot q)/(|p|\,|q|)$. Further recall that the trigonometric function cosine transforms an angle into a number between -1 and $+1$, with $\cos\alpha$ being close to 1 when the angle between p and q is very small.

Requirements

Your output will depend on your choice of hash function and several parameters, so I cannot provide sample output to help you evaluate

the correctness of your code. You probably need a few lines of code for basic operations on Euclidean vectors (length, angle). That part you should test, even if you didn't write it yourself.[3]

[3] Actually, *in particular* if you didn't write it yourself.

Deliverables

1. The source code for your implementation
2. A report in PDF. Use the report skeleton on the next page.

Background and further reading

Is this really how it's done? Well, close enough. In particular, techniques for comparing documents (for example to detect near-duplicates for web search engine reporting, data mining, or fraud dectection) are based on comparing hash values. The details are slightly different—if you want to read up on this, start with Wikipedia's article on min-wise independent hashing.

Why don't we just compute the profiles of the k-grams themselves, instead of their hash values? The problem is that the profiles become too large. Assume for a moment that there are only 24 letters in the alphabet. Then there are 24^k different k-grams, and your profile vector p would be 24^k-dimensional. Good luck storing that on your machine for $k = 20$, say! Also, most of the entries in p would end up being 0 anyway. With hashing, you need only d dimensions. This is a prototypical hashing application: Avoid storing a sparse table by hashing it into a much smaller table without losing too much information.

For comparing protein sequences, one normally uses a more precise distance estimate called the *Levenshtein* distance, sometimes called *edit distance*. For that particular distance there's actually a better algorithm called *dynamic programming*, which works fast enough for such small inputs. The same idea is used by your word processor's spell checker. This is the basis of another programming exercise that you may be exposed to in another course.

Gorilla–Sea Cucumber Hash Report

by Alice Cooper and Bob Marley[4]

[4] Complete the report by filling in your names, filling in the parts marked [...] and changing other parts wherever necessary. (For instance, the numbers is the tables are nonsense right now.) Remove the sidenotes in your final hand-in.

Results

The following table gives the similarity between each pair of species as a number between 0 and 1, higher values meaning "more similar." We have used the hash function $h(S) = [\ldots]$ with $d = [\ldots]$ and k-grams of length $k = [\ldots]$. As can be seen, the species closest to us is the [...].

	Human	Gorilla	Monkey	Horse	Deer	Pig	Cow	Gull	Trout	R. Cod	Lamprey	Sea Cuc.
Human	1	0.534	[...]									
Gorilla												
[...]												

Tests

Our static method `double cos_angle(int[] p, int[] q)` computes the cosine of the angle of two vectors of the same length d. We have tested it on the following examples:

p	q	d	value returned
$(0,1)$	$(0,1)$	2	1
$(0,1)$	$(0,2)$	2	1
$(0,1)$	$(1,0)$	2	[...]
$(0,1)$	$(0,-1)$	2	[...]
$(0,\ldots,0,1)$	$(1,0,\ldots,0)$	1000	[...]
[...]			

Similarly, our static method `length(int[] p)` computes [...].

Performance

Our python7 implementation took 3 years to compute the above on our favorite computer.