NPCNameWriteUp

December 1, 2016

Contents

Milind Luthra

1 Introduction

Github: `https://github.com/milindl/NPC-name`

This is a generator for NPC(Non-Playing Character) names. This was motivated by the book Caves of Steel, in which there's a character called Daneel. It's similar to Daniel, but not the same, and hence it's a suitable, mysterious-yet-forgettable name - perfect for NPCs.

The basic idea behind it is: take a "seed" name, and try to come up with names "close" to the seed name. A genetic algorithm comes up with new names in each generation, and a fitness function determines how close that is to the seed name. As soon as it's close enough, the genetic algorithm terminates, and we can use the multitude of names generated.

The names themselves must be in IPA. For further reference, read `https://en.wikipedia.org/wiki/International_Phonetic_Alphabet`. It's also worth looking at the chart of 44 IPA I've used, since those must be used to spell names - `http://www.antimoon.com/resources/phonchart2008.pdf`,

2 Working

2.1 Genetic Algorithm

The GA used is pretty straightforward. Except for not being binary, the implementation is not that different from usual. Interested readers can check out `https://en.wikipedia.org/wiki/Genetic_algorithm` and then try doing `https://www.codewars.com/kata/binary-genetic-algorithms/javascript` if they wish to.

The fitness I've used is

$$f = \frac{\text{AlignmentScore}(\text{generated}, \text{seed})}{\text{Mean}(\text{generated.length}, \text{seed.length})}$$

I've described the AlignmentScore below.

Note : In this working, the fitness can sometimes exceed the perfect fitness which is 1 (perfect fitness := when generated and seed are same). This is because I'm dividing with the mean, and not the max, of the lengths(the lengths are variable).

2.2 Alignment Scoring

Each possible aligment of the sequences(generated and seed) is given a score as per the following criteria:

1. 1 point for match
2. x points for mismatch
3. A penalty Δ for gaps

   x is decided by how similar sounding the alphabet in the first sequence is to the second one.

   For example, the mismatch (au, Ou) has a higher score than (t, s), since it makes a similar sound. The complete list of scores can be found in the comparison matrix, a 44x44 matrix that provides the key to mismatch scores.

   The alignment score that is the best amongst all possible alignment scores is returned by AlignmentScore.

The idea for this was motivated by sequence matching of protiens. The idea for a comparison matrix came from PAM matrices(`https://en.wikipedia.org/wiki/Point_accepted_mutation`), a system that tells us how likely a protien is to change from one form to another over time. I've tried to do the same to some extent(looking up the evolution of language), but got bored and as of now, comparison matrix is simply based on how similar the phonetics sound.

The 44 rows/cols of the matrix represent IPAs, and if we want to see how similar IPA1 is to IPA2, we can simply go to the row corresponding to IPA1 and the col corresponding to IPA2 and look at its value.

The actual algorithm itself is similar to finding pairwise sequence alignments between protien sequences. Look up `https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm` for an exact description of the algorithm.

3 Illustration

Seed name: d ei v i d (DAVID)

Generated names:

- d ei o: i d (DAOUID) (DAY - OO - EE - D)
- d ei v i TH (DAVITH)
- d ei v TH (DAVTH) (pretty much useless)
- d ei v i e:(r) (DAVIER)
- TH ei v i (THAYVI) (TH as in THIN)

Clearly, not all of them are useable, but they're not useless either.

4 Justification and Future Work

4.1 Justification for Techniques Used

I wanted to implement a GA properly, and I am interested in sequence alignment, so this seemed like a great opportunity. I could alternatively made substitution tables and then substituted in the original (ie approach it from the other side).

This would have probably been faster, but it would've lacked the feel of the language somehow changing. Though I didn't base the comparison off the actual evolution of the language, it could possibly be done by not making the comparison matrix symmetric (to model the flow of time) and by basing the data off actual fact :)

4.2 Future Work

1. Base off actual data (comparison matrix improvements)

2. Make the alignment score dependant on the *neighbouring* letters as well. Some letters tend to be in pairs, and that's ignored here.