Gibberish

A bit-sequence-to-phrase translator

Translates bit sequences

The idea is completely stolen from asana:

http://blog.asana.com/2011/09/6-sad-squid-snuggle-softly/

`[10100100011000000111101000100110]`

<->

“6 sad squid snuggle softly”

Why translate bit sequences?

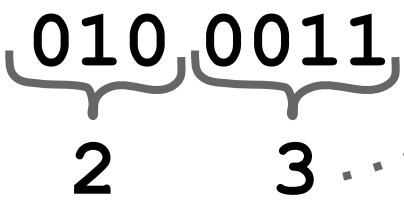
- Asana uses it for their “unique error phrases”
- Could also use it to generate passphrases
- Or just generate random sentences for fun

Unique error phrases

- When a partner makes an API call that fails, we return an error message containing an *error ID*.
- This ID is extremely useful for finding the cause of the error internally in the platform.
- The ID is not very readable for humans however as it is just a sequence of bits. So...
- **Let’s translate it!**

How does it work?

Bit sequence is divided up into smaller sequences.



These smaller bit sequences are interpreted as binary numbers.

The numbers are then used as indices in long lists of words, and the indexed words are combined.

Fancy features

- Configurability
- Optimized bit distribution
- Checksum check

Configurability

- Put some words in files, and define a schema.
- Read the schema and create a translator.
- Input some byte arrays and output some phrases.

```
files:
  animals_1:
    path     : "examples/animals.txt"
  animals_2:
    path     : "examples/more_animals.txt"
  adverbs:
    path     : "examples/adverb_of_manner.txt"
  ...
providers:
  animals:
    files : ["animals_1", "animals_2"]
  adverbs:
    files : ["adverb"]
  ...
translators:
  animal_sentence:
    format          : ["", " ", " ", " ", " ", ""]
    providers       : ["animals", "adverbs", ...]
    number_of_bits  : 32
```

Optimized bit distribution

- It could be tricky to determine how many bits from the sequence each word list is assigned. E.g. if a word list with only 6 words are assigned 3 bits, an index `[111]` = 7 is not on that list.
- An optimizer can therefore be used to optimize bit distribution for the minimum total mean word length.
- The algorithm used is a form of steepest descent with equality constraints. (total number of bits is fixed)

Checksum check

- In order to ensure that translation from bits to phrase (and vice versa) will never change, the files with words must never change.
- An optional checksum for a translator can therefore be provided to ensure that word lists have not been tampered with.

Give it a try and be creative

To translate long bit sequences you must use long lists of words, and/or many lists, to cover all those bits. Time to be creative!

```
animal_sentence_long:
  format    : ["", " ", " ", " ", " and ", " ", " ", " ", " ", " ", " in ", ""]
  providers : ["numbers", "adjectives", "identity", "animals", "numbers", "adjectives",
               "identity", "animals", "verbs", "adverbs", place]
  number_of_bits   : 58
```

“26 pale jewish ducks and 23 shy hindu swallows fly fast in sweden”
“7 sad american apes and 4 classy muslim spiders talk kookily in antarctica”
“30 mean christian cows and 23 tame chinese narwhals knit keenly in hawaii”
“15 filthy american cobras and 31 fast mexican lemurs breed eagerly in london”
...

Getting started

- Checkout the project:

  `kerrit p c experiments/gibberish`

- Load the example schema, create a translator, and give it some random input.

```
SchemaReader schemaReader =
    SchemaReader.fromResource("examples/schema_examples.yml");
PhraseTranslator translator =
    schemaReader.getTranslators().get("animal_sentence_long");
System.out.println(translator.fromLong(
    Math.abs(new Random.nextLong())));
```