ADVENTURES

IN

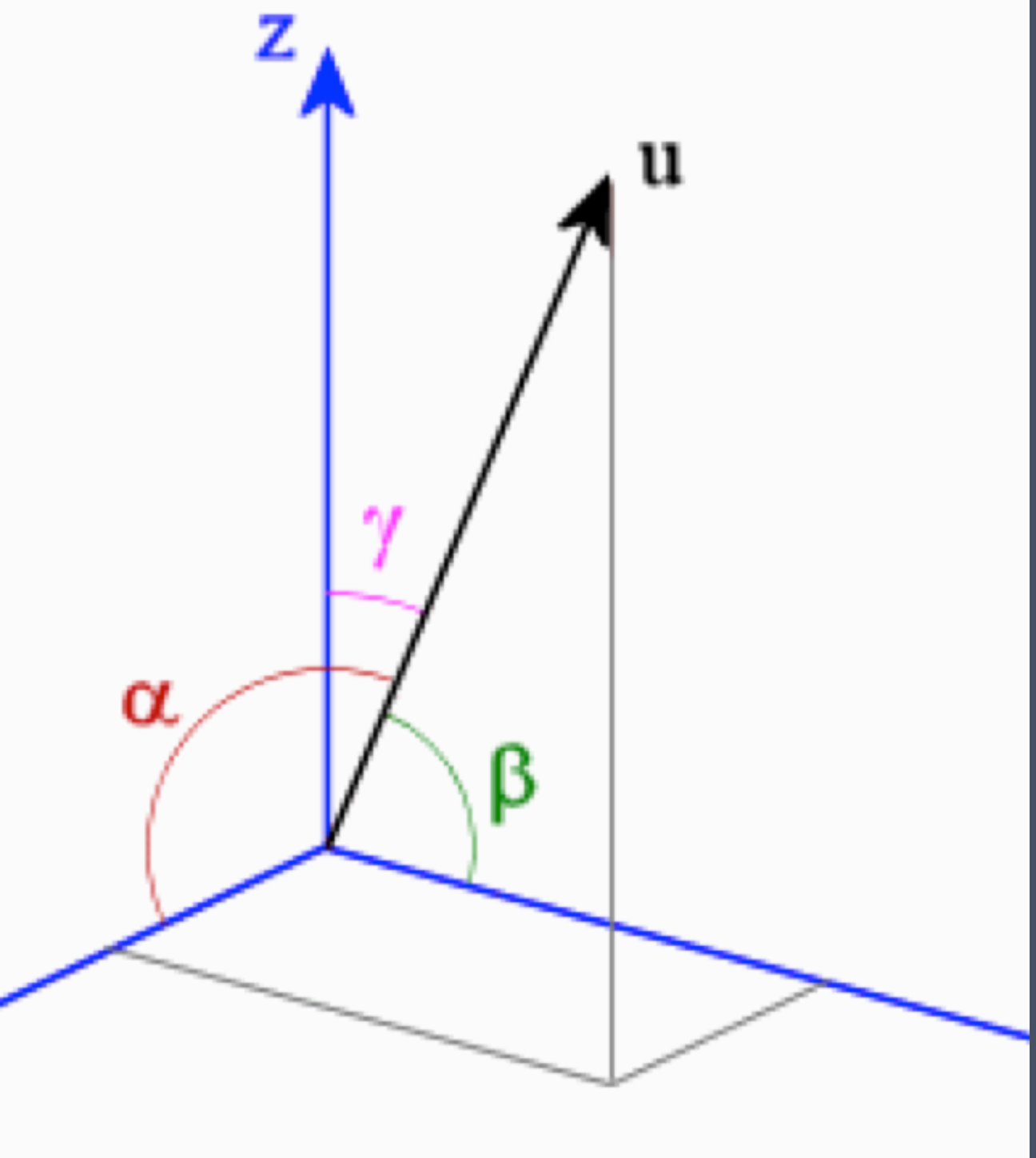
HYPERDIMENSIONAL

SPACE

WITH WORD VECTORS AND CLOJURE

BY CARIN MEIER @GIGASQUID

WHAT IS A HYPERDIMENSION?



A VECTOR HAS DIMENSIONS

[0 1 0]

A HYPERDIMENSIONAL

VECTOR HAS TONS MORE

LIKE 10,000 OR 100,000

HYPERVECTOR = HYPERDIMENSIONAL VECTOR

SOUNDS COOLER ANYWAY

VECTORS COULD BE MADE UP

OF ANYTHING REALLY...

[-1.33 23.33 99.44]

BUT IN OUR CASE. WE ARE ONLY GOING TO USE ZEROS AND ONES


THE VECTORS WILL ALSO BE SPARSE
WHICH MEANS, THEY ARE MOSTLY GOING TO BE EMPTY WITH ZEROS

SPARSE VECTORS

ALLOWS COMPUTING IN A REASONABLE AMOUNT OF TIME

CORE.MATRIX

- CLOJURE LIBRARY THAT HANDLES MATRIXES AND SPARSE VECTORS
- PROVIDES FAST DOUBLE PRECISION VECTOR MATH

(WHICH WE WILL USE EVEN THOUGH WE'RE USING ZEROS AND ONES)

VECTORZ

```
(ns hyperdimensional-playground.core
  (:require [clojure.core.matrix :as m]
            [clojure.core.matrix.linear :as ml]))

(m/set-current-implementation :vectorz)
```

GENERATE RANDOM HYPERVECTOR

```
(def sz 100000)

(defn rand-hv []
  (let [hv (m/new-sparse-array [sz])
        n (* 0.1 sz)]
    (dotimes [i n]
      (m/mset! hv (rand-int sz) 1))
    hv))
```

LET'S MAKE SOME!

```
(def a (rand-hv))
(def b (rand-hv))
(def c (rand-hv))
(def d (rand-hv))

a ;=> #vectorz/vector Large vector with shape: [100000]
```

IF RANDOM HYPERVECTORS WERE SOCKS ....

HYPERSOCKS WILL ~NEVER MATCH

NEVER?

WELL. MATHEMATICALLY SPEAKING ....

NEVER?

- PROBABILITY DISTRIBUTION
- RANDOM HYPERVECTOR WILL BE 100 STD FROM ANOTHER ONE

FOR PRACTICAL PUPOSES YOU WILL RUN OUT OF TIME BEFORE YOU BEFORE YOU RUN OUT OF UNRELATED VECTORS.

ANY TWO HYPERSOCKS/HYPERVECTORS WILL NEVER MATCH

WAIT.

HOW DO YOU TELL HOW SIMILAR TWO HYPERVECTORS ARE TO ONE ANOTHER?

COSINE SIMILARITY

USES THE DOT PRODUCT OF THE TWO VECTORS

```
(defn cosine-sim [v1 v2]
  (/ (m/dot v1 v2)
     (* (ml/norm v1) (ml/norm v2))))
```

COSINE SIMILARITY

GIVES A MEASURE FROM -1 TO 1

1 IS THE SAME

```
(cosine-sim a a) ;=>  1.0
(cosine-sim d d) ;=> 1.0
```

COSINE SIMILARITY

GIVES A MEASURE FROM -1 TO 1

~0 IS THE UNRELATED

```
(cosine-sim a b) ;=>  0.0859992468320239
(cosine-sim b c) ;=> 0.09329186588790261
(cosine-sim a c) ;=> 0.08782018973001954
```

MATH WITH HYPERVECTORS

SUM MEAN VECTOR

```
(defn mean-add [& hvs]
  (m/emap #(Math/round (double %))
   (m/div (apply m/add hvs) (count hvs))))
```

SUM MEAN VECTOR

- X = A + B
- X IS SIMILAR TO A
- X IS SIMILAR TO B

SUM MEAN VECTOR

$$X = A + B$$

```
(def x (mean-add a b))
(cosine-sim x a) ;=> 0.7234734658023224
(cosine-sim x b) ;=> 0.7252586504505658
```

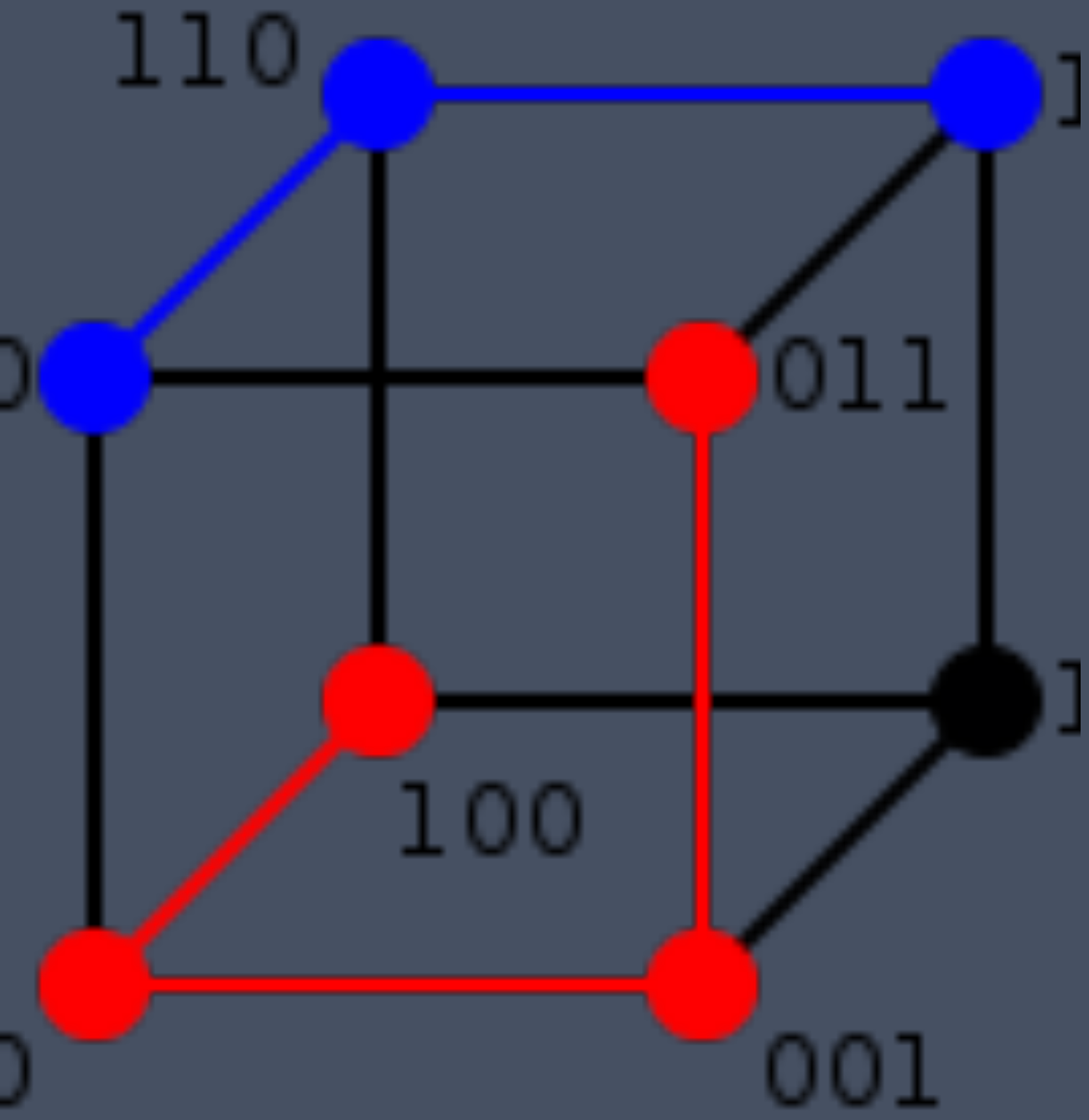
MULTIPLICATION

WITH 1S AND 0S - WE CAN USE XOR AND MOD 2

```
(defn xor-mul [v1 v2]
  (->> (m/add v1 v2)
       (m/emap #(mod % 2))))
```

HAMMING DISTANCE

NAMED AFTER RICHARD HAMMING **– 1950 PAPER** ERROR DECTECTING AND ERROR CORRECTING CODES



- USED IN TELECOMMUNICATION TO COUNT THE NUMBER OF FLIPPED BITS IN A FIXED LENGTH BINARY WORD
- ESTIMATE ERRORS
- IMPORTANT MEASURE IN ERROR DETECTION AND ERROR CORRECTION

HAMMING DISTANCE

SUM OF ALL THE XOR MULTIPLIED ELEMENTS

```
(defn hamming-dist [v1 v2]
  (m/esum (xor-mul v1 v2)))

(hamming-dist [1 0 1] [1 0 1]) ;=> 0
(hamming-dist [1 0 1 1 1 0 1] [1 0 0 1 0 0 1]) ;=> 2
(hamming-dist a a) ;=> 0
```

COOL TRICKS WITH HYPERVECTORS

- XOR MOVES (OR RANDOMIZED) THE HYPERVECTOR IN SPACE
- BUT IT PRESERVES THE DISTANCE BETWEEN TWO HYPERVECTORS

$$XA = X * A$$
$$YA = Y * A$$

HAMMING DISTANCE X AND Y = HAMMING DISTANCE OF XA AND YA

MULTIPLICATION RANDOMIZES BUT PRESERVES THE DISTANCE

```
(def x (rand-hv))
(def y (rand-hv))
(def xa (xor-mul x a))
(def ya (xor-mul y a))
(hamming-dist xa ya) ;=> 1740.0
(hamming-dist x y) ;=> 1740.0
```

HYPERVECTORS AS MAP PAIRS

YOU CAN BIND SYMBOLS TO HYPERCTORS

AND THEN RETREIVE THEM

WITH VECTOR MATH!

DATA RECORD WITH BOUND PAIRS

```
(def x (rand-hv)) ;; favorite-sock
(def y (rand-hv)) ;; cute-animal
(def z (rand-hv)) ;; name
(def a (rand-hv)) ;; red-plaid
(def b (rand-hv)) ;; duck
(def c (rand-hv)) ;; gigasquid
```

CREATE A SUM HYPERVECTOR

$$H = X * A + Y * B + Z * C$$

```
(def h (mean-add
          (xor-mul x a)
          (xor-mul y b)
          (xor-mul z c)))
```

NOW WE CAN UNBIND VALUES FROM THE SUM VECTOR

AND FIND THE CLOSEST MATCH FROM OUR KNOWN HYPERVECTORS

```
(hamming-dist a (xor-mul x h)) ;=> 1462.0  ;; closest to "red-plaid"
(hamming-dist b (xor-mul x h)) ;=> 1721.0
(hamming-dist c (xor-mul x h)) ;=> 1736.0
```

WITH COSINE SIMILARITY

```
(cosine-sim a (xor-mul x h)) ;=> 0.3195059768353112 ;; closest to "red-plaid"
(cosine-sim b (xor-mul x h)) ;=> 0.1989075567830733
(cosine-sim c (xor-mul x h)) ;=> 0.18705233578983288
```

HYPERVECTOR RECAP

- ANY TWO RANDOM VECTORS ARE FOR PRACTICAL PURPOSES UNRELATED
- YOU CAN MEASURE THE DISTANCE BETWEEN TWO HYPERVECTOR
- THE SUM OF 2 HYPERVECTORS IS SIMILIAR TO THE COMPONENTS
- XOR MULTIPLYING MOVES VECTORS INTO A DIFFERENT SPACE. BUT PRESERVES DISTANCE
- YOU CAN BIND/ UNBIND HYPERVECTOR VALUES WITH PAIRS

COGNITIVE COMPUTING

HYPERVECTORS ARE THE FOUNDATION OF EXCITING ADVANCES

WORD VECTORS

WHAT IS A WORD VECTOR?

ALSO KNOWN AS A CONTEXT VECTOR

EXPRESS WORDS WITH HYPERVECTORS

WHAT IS A WORD VECTOR?

YOU CAN COMPARE TWO WORDS

YOU CAN EVEN PERFORM REASONING WITH THEM!

FIRST WE NEED DATA

DOWNLOAD TEXT FROM 10 FAIRYTALE BOOKS ON GUTENBERG

FREQUENCY MATRIX

THE GOAL IS TO CREATE MATRIX OF HOW OFTEN A WORD APPEARS IN A DOCUMENT

ROWS: NOUNS
COLUMNS: COUNT OF WHETHER OR NOT THE WORD APPEARS

NOTE THE COUNTS ARE JUST ONES OR ZEROS

FREQUENCY MATRIX

SIZE IS BIG ENOUGH FOR HYPERDIMENSIONS

SMALL ENOUGH FOR FAST COMPUTATION

NOUNS X 10,000

GATHER YE NOUNS

USING STANFORD CORENLP

COLLECT THE NOUNS IN ALL OUR BOOKS

2500

MATRIX SIZE = 2500 X 10,000

DOWN TO CODE

- CREATE NOUN INDEX TO ROW
- CREAT SPARSE FREQ MATRIX

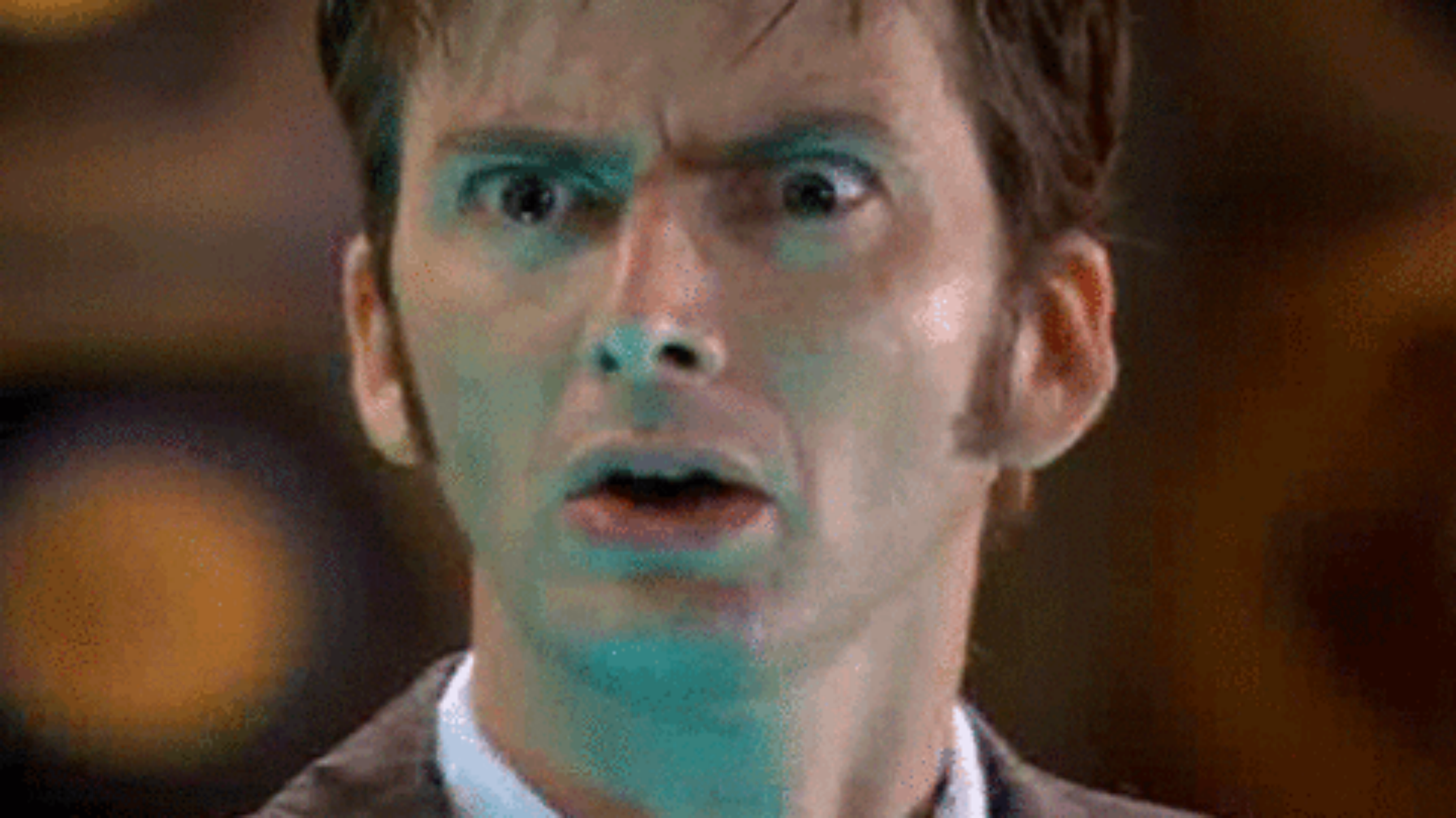
```clojure
(ns hyperdimensional-playground.context-vectors
  (:require [clojure.core.matrix :as m]
            [clojure.core.matrix.linear :as ml]
            [clojure.string :as string]
            [hyperdimensional-playground.core :refer [rand-hv cosine-sim mean-add inverse xor-mul]]
            [hyperdimensional-playground.fairytale-nouns :refer [fairy-tales-nouns book-list]]))

(m/set-current-implementation :vectorz)
;; size of the hypervectors and freq matrix columns
(def sz 10000)
;; The nouns come from a sampling of Grimm's fairy tale nouns these will
;; make up the rows in the frequency matrix
(def noun-idx (zipmap fairy-tales-nouns (range)))
(def freq-matrix (m/new-sparse-array [(count fairy-tales-nouns) sz]))
```

PROCESS A BOOK

- READ THE BOOK TEXT
- SPLIT IT INTO DOCUMENTS
- UPDATE THE FREQ MATRIX

RANDOM INDEXING

WE DON'T NEED TO WORRY ABOUT HAVING A COLUMN FOR EACH DOCUMENT



RANDOM INDEXING

BECAUSE HYPERDIMENSIONS

WE CAN RADOMLY ASSIGN 10 COLS FOR EACH DOC

UPDATE MATRIX

```
(defn process-book
  "Load a book and break it into sentence like documents and update the frequency matrix"
  [book-str]
  (let [book-text (slurp book-str)
        docs (partition 25 (map string/lower-case
                                (string/split book-text #"\s|\.|\,|\;|\!|\?")))
        doc-count (count docs)]
    (println "processing:" book-str "(with" doc-count "docs)")
    (doall (map-indexed (fn [idx doc]
                          (when (zero? (mod idx 1000)) (println "doc:" idx))
                          (update-doc! doc))
                        docs))
    (println "DONE with " book-str)))
```

RUN THE WHOLE THING

```
(doseq [book book-list]
    (process-book book))
```

ABOUT 3 SEC


EXCELLENT
NOW WHAT CAN WE DO?

HOW CLOSE ARE TWO WORDS?

COSINE SIMILARITY

```
(defn wv [word]
  "Get a hypervector for the word from the frequency matrix"
  (let [i (get noun-idx word)]
    (assert (not (nil? i)) (str word " not found"))
    (m/slice freq-matrix i)))
```

A HANDY FUNCTION FOR AN INFO MAP

```
(defn compare-wvs
  "Compare two words and give the cosine distance info map"
  [word1 word2]
  (let [wv1 (wv word1)
        wv2 (wv word2)]
    (when (not= word1 word2)
      {:word1 word1
       :word2 word2
       :cosine (cosine-sim wv1 wv2)})))
```

WORDS CLOSE TO KING

```
(sort-by :cosine[(compare-wvs "king" "queen")
                 (compare-wvs "king" "prince")
                 (compare-wvs "king" "princess")
                 (compare-wvs "king" "guard")
                 (compare-wvs "king" "goat")])
  ;; ({:word1 "king", :word2 "goat", :cosine 0.1509151478896664}
  ;;  {:word1 "king", :word2 "guard", :cosine 0.16098893367403827}
  ;;  {:word1 "king", :word2 "queen", :cosine 0.49470535530616655}
  ;;  {:word1 "king", :word2 "prince", :cosine 0.5832521795716931}
  ;;  {:word1 "king", :word2 "princess", :cosine 0.5836922474743367})
```

ADDITION

HOW CLOSE ARE BOY AND KING?

```
(cosine-sim (wv "boy") (wv "king"))
;=> 0.4299639714225314 5
```

BOY + GOLD = KING

```
(cosine-sim (mean-add (wv "boy") (wv "gold"))
            (wv "king"))
;=> 0.5876251031366048
```

BOY + GIANT = JACK

```
(cosine-sim (wv "boy") (wv "jack")) ;=> 0.33102858702785953
;; boy + giant = jack
(cosine-sim (mean-add (wv "giant") (wv "boy"))
                (wv "jack"))
 ;=>0.4491473187787431
```

FROG + PRINCESS = PRINCE

```
;;; frog + princess = prince
  (cosine-sim (wv-add "frog" "princess") (wv "prince"))
   ;=> 0.5231641991974249
```

SUBTRACTION

QUEEN = (KING - MAN) + WOMAN

```
;;; queen= (king-man) + woman
  (cosine-sim (wv "queen")
              (mean-add (wv "woman") (wv-subtract "king" "man")))
 ;=>0.5659832204544486
```

REASONING

REMEMBER MAKING PAIRS WITH XOR MULTIPLACTION?

DATABASE AS A HYPERVECTOR VALUE!

HANSEL IS THE BROTHER OF GRETEL

```
;; hansel is the brother of gretel
;; B*H + B*G
(def hansel-brother-of-gretel
    (mean-add
     (xor-mul (wv "brother") (wv "hansel"))
     (xor-mul (wv "brother") (wv "gretel"))))
```

JACK IS THE BROTHER OF HANSEL

```
(def jack-brother-of-hansel
  (mean-add
     (xor-mul (wv "brother") (wv "jack"))
     (xor-mul (wv "brother") (wv "hansel"))))
```

ADD FACTS TOGETHER

```
(def facts (mean-add hansel-brother-of-gretel
                     jack-brother-of-hansel))
```

IS JACK THE BROTHER OF GRETEL?

```
(cosine-sim
  (wv "jack")
  (xor-mul (mean-add (wv "brother") (wv "gretel"))
           facts))
;=>0.8095270629815969
```

YES

IS CINDERELLA THE BROTHER OF GRETEL?

```
(cosine-sim
  (wv "cinderella")
  (xor-mul (mean-add (wv "brother") (wv "gretel"))
           facts))
 ;=>0.1451799916656951
```

NO

IS JACK THE BROTHER OF GRETEL?

```
(cosine-sim
   (wv "jack")
   (xor-mul (mean-add (wv "brother") (wv "gretel"))
            facts))
 ;=> 0.8095270629815969
```

YES

INVENTING NEW WORDS / RELATIONS

SIBLINGS

```
(def siblings (rand-hv))
```

DEFINE SIBLING

IN TERMS OF OTHER WORD VECTORS

```
(def siblings-brother-sister
    (mean-add (xor-mul siblings (wv "brother"))
        (xor-mul siblings (wv "sister"))))
```

MORE FACTS

ADD TO OUR HYPERVECTOR DATABASE VALUE

GRETEL IS A SISTER OF HANSEL

```
;; gretel is the sister of hansel
  ;; S*G + S*H
  (def gretel-sister-of-hansel
    (mean-add
     (xor-mul (wv "sister") (wv "gretel"))
     (xor-mul (wv "sister") (wv "hansel"))))
```

GRETEL IS A SISTER OF JACK

```
;; gretel is the sister of jack
 ; S*G + S*H
 (def gretel-sister-of-jack
   (mean-add
    (xor-mul (wv "sister") (wv "gretel"))
    (xor-mul (wv "sister") (wv "jack"))))
```

NOW WE CAN ASK IT QUESTIONS

ARE HANSEL AND GRETEL SIBLINGS?

```
(cosine-sim
   (mean-add (wv "hansel") (wv "gretel"))
   (xor-mul siblings facts))
    ;=>0.627015379034067
```

YES

ARE JOHN AND ROLAND SIBLINGS?

```
(cosine-sim
   (mean-add (wv "roland") (wv "john"))
   (xor-mul siblings facts))
   ;=> 0.1984017637065277
```

NO

ARE JACK AND HANSEL SIBLINGS?

```
(cosine-sim
    (mean-add (wv "jack") (wv "hansel"))
    (xor-mul siblings facts))
   ;=>0.48003572523507465
```

YES

INTERESTING THOUGHT

WE COULD RETRACT FACTS WITH SUBTRACTION

TRIED THIS AND THE LOST TOO MUCH RESOLUTION

NEED A BETTER MEAN ADD

CONCLUSION

HYPERVECTORS ARE POWERFUL

- RESONING WITH SIMPLE VECTOR MATH
- STORE INFORMATION AND RETRIEVE IT
- FLEXIBLE COLLECTION WITH RANDOM INDEXING

CONCLUSION

KEY TO COGNITIVE COMPUTING OR AI?

REFERENCES

https://github.com/gigasquid/hyperdimensional-playground

http://redwood.berkeley.edu/pkanerva/papers/kanerva09-hyperdimensional.pdf