**Introduction**

This post walks through corpus-based methods for evaluating the efficacy of vector space models in capturing semantic relations. Here we consider the standard evaluation tool for VSMs: the **offset method for solving word analogies**. While this method is not without its limitations/criticisms

The nuts/bolts of these types of evaluations can often be glossed over in the NLP literature; here we unpack methods and work through some reproducible examples. Ultimately, our goal is to understand how standard VSM parameters (eg, dimensionality & window size) affect model efficacy, specifically for personal and/or non-standard corpora.

**Corpus & model**

We have sampled the full PG corpus to create a more manageable sub-corpus of ~7K texts and 250M words.

library(tidyverse)

setwd('/home/jtimm/jt\_work/GitHub/data\_sets/project\_gutenberg')

corpus <- readRDS('sample-pg-corpus.rds')

The type of vector space model (VSM) implemented is a GloVe model; the R package text2vec is utilized to construct this semantic space. Below, two text2vec data primitives are created: an iterator and a vectorized vocabulary.

t2v\_itokens <- text2vec::itoken(unlist(corpus),

preprocessor = tolower,

tokenizer = text2vec::space\_tokenizer,

n\_chunks = 1,

ids = names(corpus))

vocab1 <- text2vec::create\_vocabulary(t2v\_itokens, stopwords = tm::stopwords())

vocab2 <- text2vec::prune\_vocabulary(vocab1, term\_count\_min = 50)

vectorizer <- text2vec::vocab\_vectorizer(vocab2)

**Evaluation & analogy**

We use two sets of analogy problems for VSM evaluation: the standard Google data set (Mikolov et al. [2013](https://jtimm.net/2020/07/26/evaluating-vector-space-models-with-word-analogies/#ref-mikolov2013efficient)) and the BATS set (Gladkova, Drozd, and Matsuoka [2016](https://jtimm.net/2020/07/26/evaluating-vector-space-models-with-word-analogies/#ref-gladkova2016analogy)).

questions\_file <- paste0(analogy\_dir, 'questions-words.txt')

questions\_file2 <- paste0(analogy\_dir, 'bats-questions-words.txt')

google\_analogy\_set <- text2vec::prepare\_analogy\_questions(

questions\_file\_path = questions\_file,

vocab\_terms = vocab2$term)

## INFO [20:30:08.056] 11779 full questions found out of 19544 total

bats\_analogy\_set <- text2vec::prepare\_analogy\_questions(

questions\_file\_path = questions\_file2,

vocab\_terms = vocab2$term)

## INFO [20:30:08.502] 39378 full questions found out of 56036 total

tests <- c(google\_analogy\_set, bats\_analogy\_set)

**Long & short** of the vector offset method applied to analogy problems. Per some analogy defined as (1) below:

(1) a:a\* :: b:\_\_

where a = Plato, a\* = Greek, and b = Copernicus, we solve for b\* as

(2) b\* = a\* - a + b

based on the assumption that:

(3) a\* - a = b\* - b

In other words, we assume that the vector offsets between two sets of words related semantically in similar ways will be consistent when plotted in 2d semantic space. Solving for b\*, then, amounts to identifying the word whose vector representation is most similar (per cosine similarity) to a\* – a + b (excluding a\*, a, or b).

**Experimental set-up**

**Parameters**

So, to evaluate effects of window size and dimensionality on the efficacy of a GloVe model in solving analogies, we build a total of 50 GloVe models – ie, all combinations of window sizes *3:12* and model dimensions in (*50, 100, 150, 200, 250*).

p\_windows <- c(3:12)

p\_dimensions <- c(50, 100, 150, 200, 250)

ls <- length(p\_windows) \* length(p\_dimensions)

z = 0

results <- list()

details <- vector()

**Flow**

The nasty for loop below can be translated into layman’s terms as: for window size *j* and dimensions *k*, (1) build GloVe model *j-k* via text2vec::GlobalVectors, and then (2) test accuracy of GloVe model *j-k* via text2vec::check\_analogy\_accuracy.

for(j in 1:length(p\_windows)) {

tcm <- text2vec::create\_tcm(it = t2v\_itokens,

vectorizer = vectorizer,

skip\_grams\_window = p\_windows[j])

for(k in 1:length(p\_dimensions)) {

glove <- text2vec::GlobalVectors$new(rank = p\_dimensions[k], x\_max = 10)

wv\_main <- glove$fit\_transform(tcm,

n\_iter = 10,

convergence\_tol = 0.01)

glove\_vectors <- wv\_main + t(glove$components)

res <- text2vec::check\_analogy\_accuracy(

questions\_list = google\_analogy\_set,

m\_word\_vectors = glove\_vectors)

id <- paste0('-windows\_', p\_windows[j], '-dims\_', p\_dimensions[k])

z <- z + 1

results[[z]] <- res

details[z] <- id

}

}

**Output structure**

Responses to the analogy test are summarized as a list of data frames – one for each of our 50 GloVe models.

names(results) <- details

answers <- results %>%

bind\_rows(.id = 'model')

Test components have been hashed (per text2vec) to speed up the “grading” process – here, we cross things back to actual text.

key <- vocab2 %>%

mutate(id = row\_number()) %>%

select(id, term)

tests\_df <- lapply(tests, data.frame) %>%

bind\_rows() %>%

mutate(aid = row\_number())

tests\_df$X1 <- key$term[match(tests\_df$X1, key$id)]

tests\_df$X2 <- key$term[match(tests\_df$X2, key$id)]

tests\_df$X3 <- key$term[match(tests\_df$X3, key$id)]

tests\_df$X4 <- key$term[match(tests\_df$X4, key$id)]

Then we join test & response data to create a single, readable data table.

predicted\_actual <- answers %>%

group\_by(window, dimensions) %>%

mutate(aid = row\_number()) %>%

ungroup() %>%

left\_join(key, by = c('predicted' = 'id')) %>%

rename(predicted\_term = term) %>%

left\_join(key, by = c('actual' = 'id')) %>%

rename(actual\_term = term) %>%

left\_join(tests\_df %>% select(aid, X1:X3)) %>%

na.omit %>%

mutate(correct = ifelse(predicted == actual, 'Y', 'n'))

A sample of this table is presented below. incorrect answers are generally more interesting.

**Results: model parameters**

Our performance metric for a given model, then, is the percentage of correct analogy responses, or analogy accuracy. Accuracy scores by dimensions, window size, and data set/analogy category are computed below.

mod\_category\_summary <- answers %>%

mutate(correct = ifelse(predicted == actual, 'Y', 'N')) %>%

mutate(dset = ifelse(grepl('gram', category), 'google', 'bats')) %>%

group\_by(dset, dimensions, window, category, correct) %>%

summarize(n = n()) %>%

ungroup() %>%

spread(correct, n) %>%

mutate(N = as.integer(N),

Y = as.integer(Y),)

**Vector dimensionality effect**

mod\_summary <- mod\_category\_summary %>%

filter(![is.na](http://is.na)(Y)) %>% #

group\_by(dset, window, dimensions) %>%

summarize(N = sum(N), Y = sum(Y)) %>%

ungroup() %>%

mutate (per = round(Y/(N+Y) \*100, 1))

The plot below illustrates the relationship between analogy accuracy and # of model dimensions as a function of window size, faceted by analogy set. Per plot, GloVe model gains in analogy performance plateau at 150 dimensions for all window sizes; in several instances, accuracy decreases at dimensions > 150. Also – the BATS collection of analogies would appear to be a bit more challenging.

mod\_summary %>%

ggplot() +

geom\_line(aes(x = dimensions,

y = per,

color = factor(window),

linetype = factor(window)), size = 1) +

facet\_wrap(~dset) +

theme\_minimal() +

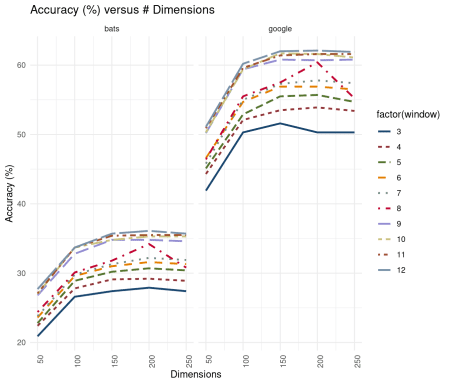
ggthemes::scale\_color\_stata() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

theme(legend.position = 'right') +

ylab('Accuracy (%)') + xlab("Dimensions") +

labs(title = 'Accuracy (%) versus # Dimensions')



**Window Size effect**

The plot below illustrates the relationship between analogy accuracy and window size as a function of dimensionality. Here, model performance improves per step-increase in widow size – accuracy seems to improve most substantially from window sizes 8 to 9. Also, some evidence of a leveling off at window sizes > 9 for higher-dimension models.

**The simplest and highest performing model**, then, for this particular corpus (in the aggregate) is a *window size = 10 and dimensions = 150* model.

mod\_summary %>%

ggplot() +

geom\_line(aes(x = window,

y= per,

color = factor(dimensions),

linetype = factor(dimensions)

),

size = 1.25) +

facet\_wrap(~dset) +

theme\_minimal() +

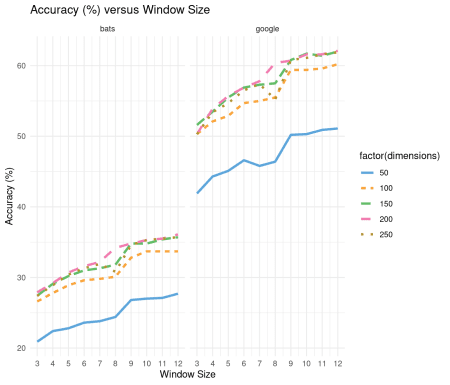
ggthemes::scale\_color\_few() +

scale\_x\_continuous(breaks=c(3:12)) +

theme(legend.position = 'right') +

ylab('Accuracy (%)') + xlab("Window Size") +

labs(title = 'Accuracy (%) versus Window Size')



**Results: analogy categories**

Next, we disaggregate model efficacy by analogy category and window size, holding dimensionality constant at 150. Results for each set of analogy problems are visualized as tiled “heatmaps” below; dark green indicates higher accuracy within a particular category, dark brown lower accuracy. Folks have noted previously in the literature that smaller window sizes tend to be better at capturing relations more semantic (as opposed to more grammatical) in nature. Some evidence for that here.

**Google analogy set**

mod\_category\_summary %>%

filter(!grepl('\_', category)) %>%

filter(dimensions == 150) %>%

mutate(per = round(Y/(N+Y) \*100, 1)) %>%

filter(![is.na](http://is.na)(per)) %>%

group\_by(category) %>%

mutate(rank1 = rank(per)) %>%

ungroup() %>%

ggplot(aes(x = factor(window), y = category)) +

geom\_tile(aes(fill = rank1)) +

geom\_text(aes(fill = rank1, label = per), size = 3) +

scale\_fill\_gradient2(low = scales::muted("#d8b365"),

mid = "#f5f5f5",

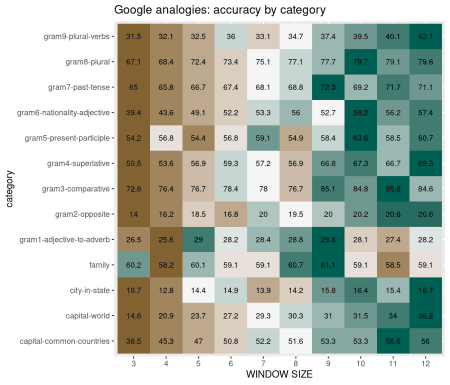
high = scales::muted('#5ab4ac'),

midpoint = 5) +

theme(legend.position = 'none') +

xlab('WINDOW SIZE') +

ggtitle('Google analogies: accuracy by category')



**BATS analogy set**

mod\_category\_summary %>%

filter(grepl('\_', category)) %>%

filter(dimensions == 150) %>%

mutate(per = round(Y/(N+Y) \*100, 1),

category = gsub('^.\*/','', category)) %>%

filter(![is.na](http://is.na)(per)) %>%

group\_by(category) %>%

mutate(rank1 = rank(per)) %>%

ungroup() %>%

ggplot(aes(x = factor(window), y = category)) +

geom\_tile(aes(fill = rank1)) +

geom\_text(aes(fill = rank1, label = per), size = 3) +

scale\_fill\_gradient2(low = scales::muted("#d8b365"),

mid = "#f5f5f5",

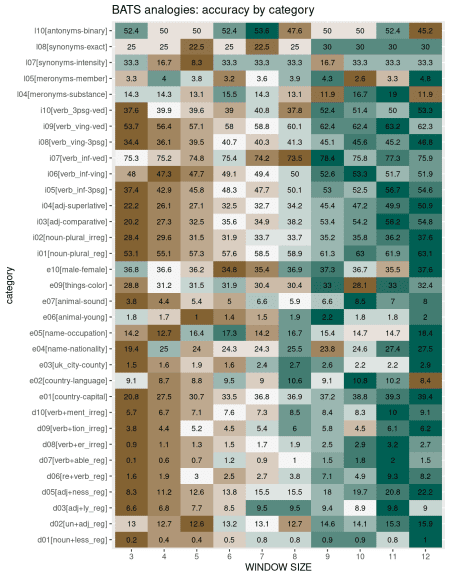
high = scales::muted('#5ab4ac'),

midpoint = 5) +

theme(legend.position = 'none') +

xlab('WINDOW SIZE') +

ggtitle('BATS analogies: accuracy by category')



**Visualizing vector offsets**

**GloVe model in two dimensions**

For demonstration purposes, we use a semantic space derived from the *window size = 5 and dimensions = 100* GloVe model. This space is transformed from 100 GloVe dimensions to two dimensions via principal component analysis.

pca\_2d <- prcomp(glove\_vectors,

scale = TRUE, center = TRUE) %>%

pluck(5) %>%

data.frame() %>%

select(PC1, PC2)

**A 30,000 foot view** of this two-dimensional semantic space.

ggplot(data = pca\_2d,

aes(x = PC1,

y = PC2)) +

geom\_point(size = .05, color = 'lightgray') +

geom\_text(data = pca\_2d,

aes(x = PC1,

y = PC2,

label = rownames(pca\_2d)),

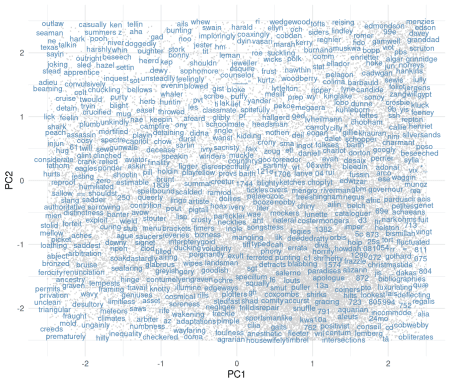
size = 3,

color = 'steelblue',

check\_overlap = TRUE) +

xlim (-2.5,2.5) + ylim(-2.5,2.5) +

theme\_minimal()



**Copernicus & Plato**

x1 = 'copernicus'; x2 = 'polish'; y1 = 'plato'; y2 = 'greek'

y <- pca\_2d[rownames(pca\_2d) %in% c(x1, x2, y1, y2),]

Finally, a visual demonstration of the vector offset method at-work in solving the **Copernicus analogy** problem. Situated within the full semantic space for context.

ggplot(data = pca\_2d,

aes(x = PC1,

y = PC2)) +

geom\_text(data = pca\_2d,

aes(x = PC1,

y = PC2,

label = rownames(pca\_2d)),

size = 3,

color = 'gray',

check\_overlap = TRUE) +

xlim(min(c(off\_dims$x1, off\_dims$x2)),

max(c(off\_dims$x1, off\_dims$x2))) +

ylim(min(c(off\_dims$y1, off\_dims$y2)),

max(c(off\_dims$y1, off\_dims$y2))) +

geom\_segment(data = off\_dims[1:2,],

aes(x = x2, y = y2,

xend = x1, yend = y1),

color = '#df6358',

size = 1.25,

arrow = arrow(length = unit(0.025, "npc"))) +

geom\_segment(data = off\_dims[3:4,],

aes(x = x1, y = y1,

xend = x2, yend = y2),

color = 'steelblue',

size = 1.25,

#linetype = 4,

arrow = arrow(length = unit(0.025, "npc"))) +

ggrepel::geom\_text\_repel(

data = y,

aes(label = toupper(rownames(y))),

direction = "y",

hjust = 0,

size = 4.25,

color = 'black') +

theme\_minimal() +

ggtitle(paste0(x1, ':', x2, ' :: ', y1, ':', y2))

**Summary & caveats**

Mostly an excuse to gather some thoughts. I use GloVe models quite a bit for exploratory purposes. To better trust insights gained from exploration, it is generally nice to have an evaluative tool, however imperfect. And certainly to justify parameter selection for corpus-specific tasks. Hopefully a useful  
resource and guide.