At the beginning of this year, Let’s get started with the [stm and tidytext packages for topic modeling](https://juliasilge.com/blog/sherlock-holmes-stm/). I have been doing more topic modeling, so I wanted to share some workflows I have found useful for

* training many topic models at one time,
* evaluating topic models and understanding model diagnostics, and
* exploring and interpreting the content of topic models.

I’ve been doing all my topic modeling with [Structural Topic Models](http://www.structuraltopicmodel.com/) and the [stm](https://cran.r-project.org/package=stm) package lately, and it has been ✨GREAT✨. One thing I am not going to cover in this blog post is how to use document-level covariates in topic modeling, i.e., how to train a model with topics that can vary with some continuous or categorical characteristic of your documents. I hope to build up some posts about that, but in the meantime, you can check out the [stm vignette](https://cran.r-project.org/web/packages/stm/vignettes/stmVignette.pdf).

**Modeling the Hacker News corpus**

In my last blog post, I demonstrated how to get started with about a book’s worth of text, which is a TEENY TINY amount of text for a topic model. This time around, I’d like to demonstrate how to go about interpreting results with a more realistic set of text, something more like what you might actually want to model topics with in the real world, so let’s turn to the [Hacker news corpus](https://cloud.google.com/bigquery/public-data/hacker-news) and download 100,000 texts using the [bigrquery](https://cran.r-project.org/package=bigrquery) package.

library(bigrquery)

library(tidyverse)

sql <- "#legacySQL

SELECT

stories.title AS title,

stories.text AS text,

FROM

[bigquery-public-data:hacker\_news.full] AS stories

WHERE

stories.deleted IS NULL

LIMIT

100000"

hacker\_news\_raw <- query\_exec(sql, project = project, max\_pages = Inf)

After we have the text downloaded, let’s clean the text and make a data frame containing only the text, plus an ID to identify each “document”, i.e., post.

hacker\_news\_text <- hacker\_news\_raw %>%

as\_tibble() %>%

mutate(title = na\_if(title, ""),

text = coalesce(title, text)) %>%

select(-title) %>%

mutate(text = str\_replace\_all(text, "'|"|/", "'"), ## weird encoding

text = str\_replace\_all(text, "", " "), ## links

text = str\_replace\_all(text, ">|<|&", " "), ## html yuck

text = str\_replace\_all(text, "&#[:digit:]+;", " "), ## html yuck

text = str\_remove\_all(text, "<[^>]\*>"), ## mmmmm, more html yuck

postID = row\_number())

Now it’s time to tokenize and tidy the text, remove some stop words (and numbers, although this is an analytical choice that you might want to try in a different way), and then cast to a sparse matrix. I’m using the token = "tweets" option for tokenizing because it often performs the most sensibly with text from online forums, such as Hacker News (and Stack Overflow, and Reddit, and so on). I used a quanteda dfm as the input to the topic modeling algorithm, but here I’m using a plain old sparse matrix. Either one works.

library(tidytext)

tidy\_hacker\_news <- hacker\_news\_text %>%

unnest\_tokens(word, text, token = "tweets") %>%

anti\_join(get\_stopwords()) %>%

filter(!str\_detect(word, "[0-9]+")) %>%

add\_count(word) %>%

filter(n > 100) %>%

select(-n)

hacker\_news\_sparse <- tidy\_hacker\_news %>%

count(postID, word) %>%

cast\_sparse(postID, word, n)

**Train and evaluate topic models**

Now it’s time to train some topic models! 💪 but in this post, we’re going to go to the next level. We’re not going to train just one topic model, but a whole group of them, with different numbers of topics, and then evaluate these models. In topic modeling, like with k-means clustering, we don’t know ahead of time how many topics we should use, and research in this area says there is no “right” answer for the number of topics that is appropriate for any given corpus. Here, let’s try a number of different values for \(K\) (the number of topics) from 20 to 100.

With 100,000 texts this modeling takes a while 😩 so I have used I have used the furrr package for parallel processing.

library(stm)

library(furrr)

plan(multiprocess)

many\_models <- data\_frame(K = c(20, 40, 50, 60, 70, 80, 100)) %>%

mutate(topic\_model = future\_map(K, ~stm(hacker\_news\_sparse, K = .,

verbose = FALSE)))

Now that we’ve fit all these topic models with different numbers of topics, we can explore how many topics are appropriate/good/“best”. The code below to find k\_result is similar to stm’s own [searchK()](https://github.com/bstewart/stm/blob/master/R/searchK.R) function, but it allows you to evaluate models trained on a sparse matrix (or a quanteda dfm) instead of only stm’s corpus data structure, as well as to dig into the model diagnostics yourself in detail. Some of these functions were not originally flexible enough to take a sparse matrix or dfm as input.

heldout <- make.heldout(hacker\_news\_sparse)

k\_result <- many\_models %>%

mutate(exclusivity = map(topic\_model, exclusivity),

semantic\_coherence = map(topic\_model, semanticCoherence, hacker\_news\_sparse),

eval\_heldout = map(topic\_model, eval.heldout, heldout$missing),

residual = map(topic\_model, checkResiduals, hacker\_news\_sparse),

bound = map\_dbl(topic\_model, function(x) max(x$convergence$bound)),

lfact = map\_dbl(topic\_model, function(x) lfactorial(x$settings$dim$K)),

lbound = bound + lfact,

iterations = map\_dbl(topic\_model, function(x) length(x$convergence$bound)))

k\_result

## # A tibble: 7 x 10

## K topic\_model exclusivity semantic\_coherence eval\_heldout residual bound lfact lbound iterations

##

## 1 20 -15991207. 42.3 -15991165. 19

## 2 40 -15990161. 110. -15990051. 26

## 3 50 -15998161. 148. -15998012. 30

## 4 60 -16014305. 189. -16014117. 33

## 5 70 -16007921. 230. -16007690. 41

## 6 80 -16018471. 274. -16018197. 48

## 7 100 -16003418. 364. -16003055. 114

We’re evaluating things like the residuals, the [semantic coherence](https://dl.acm.org/citation.cfm?id=2145462) of the topics, the likelihood for held-out datasets, and more. We can make some diagnostic plots using these quantities to understand how the models are performing at various numbers of topics. The following code makes a diagnostic plot similar to one that comes built in to the stm package.

k\_result %>%

transmute(K,

`Lower bound` = lbound,

Residuals = map\_dbl(residual, "dispersion"),

`Semantic coherence` = map\_dbl(semantic\_coherence, mean),

`Held-out likelihood` = map\_dbl(eval\_heldout, "expected.heldout")) %>%

gather(Metric, Value, -K) %>%

ggplot(aes(K, Value, color = Metric)) +

geom\_line(size = 1.5, alpha = 0.7, show.legend = FALSE) +

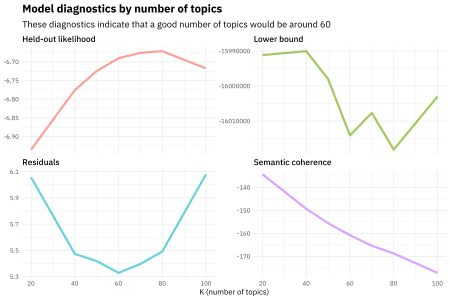
facet\_wrap(~Metric, scales = "free\_y") +

labs(x = "K (number of topics)",

y = NULL,

title = "Model diagnostics by number of topics",

subtitle = "These diagnostics indicate that a good number of topics would be around 60")



The held-out likelihood is highest between 60 and 80, and the residuals are lowest around 60, so perhaps a good number of topics would be around there.

Semantic coherence is maximized when the most probable words in a given topic frequently co-occur together, and it’s a metric that correlates well with human judgment of topic quality. Having high semantic coherence is relatively easy, though, if you only have a few topics dominated by very common words, so you want to look at both semantic coherence and exclusivity of words to topics.

k\_result %>%

select(K, exclusivity, semantic\_coherence) %>%

filter(K %in% c(20, 60, 100)) %>%

unnest() %>%

mutate(K = as.factor(K)) %>%

ggplot(aes(semantic\_coherence, exclusivity, color = K)) +

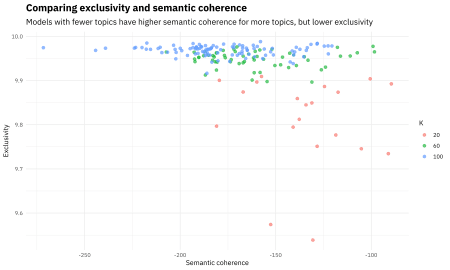
geom\_point(size = 2, alpha = 0.7) +

labs(x = "Semantic coherence",

y = "Exclusivity",

title = "Comparing exclusivity and semantic coherence",

subtitle = "Models with fewer topics have higher semantic coherence for more topics, but lower exclusivity")



So for this analysis, it looks a good choice could be the model with **60** topics.

topic\_model <- k\_result %>%

filter(K == 60) %>%

pull(topic\_model) %>%

.[[1]]

topic\_model

## A topic model with 60 topics, 98000 documents and a 3828 word dictionary.

**Explore the topic model**

We’ve trained topic models, evaluated them, and picked one to use, so now let’s see what this topic model tells us about the Hacker News corpus. In real life analysis, this process would be iterative, moving from exploring and interpreting a model back and forth to diagnostics and evaluation in order to decide how best to model a corpus. One of the reasons I embrace tidy data principles and tidy tools is that this iterative process is streamlined. For example, let’s tidy() the beta matrix for our topic model and look at the probabilities that each word is generated from each topic.

td\_beta <- tidy(topic\_model)

td\_beta

## # A tibble: 229,680 x 3

## topic term beta

##

## 1 1 arguments 8.56e-20

## 2 2 arguments 4.20e-15

## 3 3 arguments 3.21e-15

## 4 4 arguments 9.23e-13

## 5 5 arguments 1.45e-12

## 6 6 arguments 5.44e-18

## 7 7 arguments 1.04e-24

## 8 8 arguments 1.52e-11

## 9 9 arguments 4.77e-16

## 10 10 arguments 2.29e-16

## # ... with 229,670 more rows

I’m also quite interested in the probabilities that each document is generated from each topic, that gamma matrix.

td\_gamma <- tidy(topic\_model, matrix = "gamma",

document\_names = rownames(hacker\_news\_sparse))

td\_gamma

## # A tibble: 5,880,000 x 3

## document topic gamma

##

## 1 1 1 0.00631

## 2 2 1 0.00446

## 3 3 1 0.00670

## 4 4 1 0.00767

## 5 5 1 0.00742

## 6 6 1 0.00907

## 7 7 1 0.00479

## 8 8 1 0.00906

## 9 9 1 0.00801

## 10 10 1 0.00881

## # ... with 5,879,990 more rows

Let’s combine these to understand the topic prevalence in the Hacker News corpus, and which words contribute to each topic.

library(ggthemes)

top\_terms <- td\_beta %>%

arrange(beta) %>%

group\_by(topic) %>%

top\_n(7, beta) %>%

arrange(-beta) %>%

select(topic, term) %>%

summarise(terms = list(term)) %>%

mutate(terms = map(terms, paste, collapse = ", ")) %>%

unnest()

gamma\_terms <- td\_gamma %>%

group\_by(topic) %>%

summarise(gamma = mean(gamma)) %>%

arrange(desc(gamma)) %>%

left\_join(top\_terms, by = "topic") %>%

mutate(topic = paste0("Topic ", topic),

topic = reorder(topic, gamma))

gamma\_terms %>%

top\_n(20, gamma) %>%

ggplot(aes(topic, gamma, label = terms, fill = topic)) +

geom\_col(show.legend = FALSE) +

geom\_text(hjust = 0, nudge\_y = 0.0005, size = 3,

family = "IBMPlexSans") +

coord\_flip() +

scale\_y\_continuous(expand = c(0,0),

limits = c(0, 0.09),

labels = percent\_format()) +

theme\_tufte(base\_family = "IBMPlexSans", ticks = FALSE) +

theme(plot.title = element\_text(size = 16,

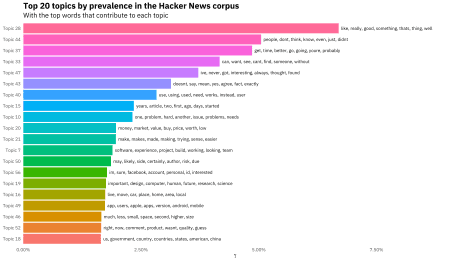
family="IBMPlexSans-Bold"),

plot.subtitle = element\_text(size = 13)) +

labs(x = NULL, y = expression(gamma),

title = "Top 20 topics by prevalence in the Hacker News corpus",

subtitle = "With the top words that contribute to each topic")



We can look at all the topics, ordered by prevalence.

gamma\_terms %>%

select(topic, gamma, terms) %>%

kable(digits = 3,

col.names = c("Topic", "Expected topic proportion", "Top 7 terms"))

| **Topic** | **Expected topic proportion** | **Top 7 terms** |
| --- | --- | --- |
| Topic 28 | 0.067 | like, really, good, something, thats, thing, well |
| Topic 44 | 0.050 | people, dont, think, know, even, just, didnt |
| Topic 37 | 0.049 | get, time, better, go, going, youre, probably |
| Topic 33 | 0.042 | can, want, see, cant, find, someone, without |
| Topic 47 | 0.037 | ive, never, got, interesting, always, thought, found |
| Topic 43 | 0.031 | doesnt, say, mean, yes, agree, fact, exactly |
| Topic 40 | 0.028 | use, using, used, need, works, instead, user |
| Topic 15 | 0.023 | years, article, two, first, ago, days, started |
| Topic 10 | 0.023 | one, problem, hard, another, issue, problems, needs |
| Topic 20 | 0.020 | money, market, value, buy, price, worth, low |
| Topic 21 | 0.020 | make, makes, made, making, trying, sense, easier |
| Topic 7 | 0.019 | software, experience, project, build, working, looking, team |
| Topic 50 | 0.019 | may, likely, side, certainly, author, risk, due |
| Topic 56 | 0.018 | im, sure, facebook, account, personal, id, interested |
| Topic 19 | 0.018 | important, design, computer, human, future, research, science |
| Topic 16 | 0.017 | live, move, car, place, home, area, local |
| Topic 49 | 0.017 | app, users, apple, apps, version, android, mobile |
| Topic 46 | 0.017 | much, less, small, space, second, higher, size |
| Topic 52 | 0.017 | right, now, comment, product, wasnt, quality, guess |
| Topic 18 | 0.016 | us, government, country, countries, states, american, china |
| Topic 51 | 0.016 | year, support, last, old, next, linux, per |
| Topic 17 | 0.015 | language, programming, c, learn, python, languages, written |
| Topic 12 | 0.014 | company, business, startup, name, ideas, early, employees |
| Topic 2 | 0.014 | part, real, reason, life, story, guy, bitcoin |
| Topic 31 | 0.014 | code, open, source, write, writing, test, program |
| Topic 22 | 0.014 | pay, cost, paid, costs, paying, rate, amount |
| Topic 9 | 0.013 | email, ask, show, best, link, please, form |
| Topic 54 | 0.013 | system, large, control, systems, built, scale, require |
| Topic 55 | 0.013 | wrong, nothing, page, difference, whats, theres, view |
| Topic 26 | 0.013 | case, law, cases, nobody, wants, serious, laws |
| Topic 53 | 0.013 | change, group, history, position, political, involved, individual |
| Topic 25 | 0.012 | read, top, list, reading, add, news, books |
| Topic 11 | 0.012 | google, internet, search, browser, ie, address, chrome |
| Topic 39 | 0.012 | standard, eg, types, implementation, object, structure, table |
| Topic 42 | 0.012 | new, create, technology, website, rules, existing, created |
| Topic 38 | 0.012 | high, school, tax, average, poor, course, kids |
| Topic 30 | 0.012 | run, running, server, api, application, client, database |
| Topic 35 | 0.012 | data, information, public, access, private, analysis, details |
| Topic 48 | 0.012 | line, available, video, tools, vs, basic, tool |
| Topic 24 | 0.012 | different, true, model, definitely, yeah, left, completely |
| Topic 41 | 0.011 | service, services, provide, customers, called, trust, customer |
| Topic 36 | 0.011 | game, play, games, fun, sound, water, playing |
| Topic 34 | 0.011 | almost, fast, extremely, field, speed, tend, theory |
| Topic 5 | 0.011 | person, must, common, wonder, situation, along, net |
| Topic 29 | 0.011 | example, type, call, result, function, lack, currently |
| Topic 59 | 0.011 | number, phone, =, +, numbers, normal, random |
| Topic 60 | 0.011 | also, just, still, already, well, way, least |
| Topic 14 | 0.011 | free, windows, full, online, key, microsoft, offer |
| Topic 3 | 0.011 | work, job, state, book, jobs, leave, hire |
| Topic 27 | 0.010 | world, security, talking, rest, parts, seeing, changed |
| Topic 13 | 0.010 | web, site, post, x, os, sites, blog |
| Topic 23 | 0.010 | power, become, benefit, society, food, energy, cars |
| Topic 32 | 0.009 | often, sometimes, word, ones, words, turn, context |
| Topic 1 | 0.009 | question, level, whether, answer, questions, asked, asking |
| Topic 57 | 0.009 | set, simple, default, complex, relatively, push, implement |
| Topic 6 | 0.009 | big, companies, many, tech, deal, industry, huge |
| Topic 45 | 0.008 | content, social, network, results, media, ads, ad |
| Topic 8 | 0.008 | file, terms, legal, files, step, purpose, license |
| Topic 4 | 0.008 | women, culture, age, men, young, people, older |
| Topic 58 | 0.006 | hn, community, light, others, reddit, come, hey |

We can see here that the first several topics are focused around general purpose English words in different categories of meaning. About 10 topics down, we see a topic about markets, money, and value. A bit below that, we see the first topic with explicitly technical-ish terms like software, build, and project. There is a topic that combined “make”, “makes”, “made”, and “making”. Notice that I did not stem these words before modeling.

So there you have it! We trained topic models at multiple values of \(K\), evaluated them, and then explored our model. Let me know if you have any questions or feedback!