Urgency Analysis - Memo 1

Henrique Sposito

2024-02-29

Urgency analysis (UA) helps extract and ranks preferences from political discourses. Urgency in political discourses encompasses the degree to which call to actions in relation to policy objects (that demand dedicated political attention) are timely (e.g. how soon) and necessary (e.g. how needed). Call to actions are internally (inner) and externally (outer) directed demands for resilience or change related to present (continued) or future political actions. For instance, when talking about climate change induced extreme weather events, a politician might say:

"Climate change is causing a higher incidence of extreme weather conditions that affect all of us, we must act now."

In this case, the call to action is exemplified by "must act" in relation to climate change related extreme weather, the policy object. The call to action is timely because the must act "now" and needed this issue is happening more frequently and affecting more people. However, how do we rank preferences in political discourse from this? Take, for example, the following similar statement:

"Climate change is causing a higher incidence of extreme weather conditions that affect many individuals, we must act soon."

The call to action is not as timely (e.g. "soon" vs. "now") or as needed (e.g. "all of us" vs. "many individuals") as beforehand. Therefore, it is less urgent than the first instance.

Politicians, however, talk about many topics over time, across settings, and even in the same speech. This means, since politicians often talk about many topic in discourses, text must first segmented into smaller coherent sections according to topics. Furthermore, observations are not only counted, but scored in relations to how timely and necessary call to action is justified to be. This requires that findings are normalized and contextualized. Moreover, political discourses do not happen in a vacuum. Rather, they have a speaker (politician), a location (where), and a time (when). All of this is important meta information to be retained as part of any analysis. In practice, the poldis R package helps users to load, clean, segment, score urgency, and visualize findings in large numbers of political discourses.

Loading and standardizing data

The first step is to load and clean the text data. We will use the a sample data of 20 presidential from the US and China speeches (in English).

```
# step 1: segment, by topic or however user choose
sample_text <- read_csv("sample_text.csv")
# How to store text data in package, as sentences, tokens or text matrix?
corp <- corpus(sample_text) |> corpus_reshape(to = "sentences") # sentences
toks <- tokens(corp, remove_punct = TRUE, remove_symbols = TRUE,
               remove_numbers = TRUE, remove_url = TRUE) # tokens
dfmt <- dfm(toks) |> dfm_remove(stopwords("en")) # document feature matrix
spacyr::spacy_initialize(model = "en_core_web_sm")
parse <-  spacyr::spacy_parse(corp) # NLP tokens
spacyr::spacy_finalize()
```

Topic Segmentation

Segment can be done in three ways, manual (e.g. choose sentences around key words), semi-automated (e.g. train a set of the data and classify texts), and automated (e.g. texttiling or LDA). However, one of the the main approach used for automated text segmentation, texttiling has not been implemented in R only in Python for now. This could provides us with an opportunity to adapt/offer this method as part of the package, but this should be carefully considered since it would require a lot of work.

First, let's try a manual approach. Assume we are interested in "climate change" as an object of policy. We can get two sentences before or after every occurrence of the words in dictionary for climate change we setup. This is, we segment text explicitly based on the topic of choice. Although efficient, this approach requires we get the "context" for all topics we are interested in before comparing them.

```
#library(poldis)
#context <- extract_context(match = "climate change|global warming|deforest",
#                           v = sample_text$text, level = "sentences", n = 1)
```

Second, let's try to classify sentences into whether they are general "call to actions". To do so, instead of a dictionary we will use a tiny training set (2 texts coded) to classify sentences.

```
training_set <- data.frame(id = rownames(data.frame(corp)), sentences = corp) |>
  filter(grepl("text9|text18", id)) |>
  mutate(call_to_action = c(0,0,0,0,0,0,0,0,0,0, 0,0,1,1,1,0,0,0,0,0,
                            1,0,0,0,0,0,1,0,0,0, 0,0,0,0,0,1,0,0,0,0,
                            0,0,1,0,0,0,0,0,1,0, 0,0,1,0,0,0,0,1,0,0,
                            1,0,1,0,0,1,1,0,0,0, 1,1,0,0,0,0,0,0,
                            0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,
                            0,0,1,0,1,1,0,1,0,0, 0,0,0,0,0,0,1,0,0,1,
                            0,0,0,1,0,0,0,0,1,0, 1,0,0,0,0,0,0,0,0,0,
                            1,1,1,0,0,0,0,0,0,0, 0,0,0,0,0,0,1,0,0,0,
                            0,0,1,0,0,1,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,
                            1,0,0,0,0,0,0,0,0,1, 0,0,0,0,0,1,1,0,0,0,
                            0,0,0,0,0,0,0,0,0,1, 0,0,0,0,0,0,0,0,0,0,
                            1,0,1,0,0,0,0,0,0,0, 1,1,1,1,1,1,1,0,0,0,
                            0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,
                            0,0,1,1,1,1,1,0,0,0, 0,0,0,0,0,0,0,0))
classify_set <- data.frame(id = rownames(data.frame(corp)), sentences = corp) |>
  filter(!grepl("text9|text18", id))
# SVM binary text classifier, for the sake of it...
clean_text <- function(v) { # clean stuff
  out <- stringi::stri_trans_general(v, 'latin-ascii')
  out <- tolower(out)
  out <- tm::removeNumbers(out)
  out <- tm::removePunctuation(out)
  out <- tm::removeWords(out, stopwords('pt'))
  out <- stringr::str_squish(out)
  out <- trimws(out)
}
training_set$sentences <- clean_text(training_set$sentences)
train_matrix <- create_matrix(training_set$sentences, weighting = weightTfIdf)
classify_set$sentences <- clean_text(classify_set$sentences)
score_matrix <- create_matrix(classify_set$sentences,
                              originalMatrix = train_matrix,
                              weighting = weightTfIdf)
model <- svm(x = train_matrix, y = as.numeric(training_set$call_to_action)) # train
pred <- predict(model, score_matrix) # predict
```

```
classify_set$call_to_action <- pred # add to dataset
arrange(classify_set, -call_to_action) |> select(sentences) |> slice_head(n = 20)
```

```
##                                                                                              sen
## text19.477                                                                       longer is there signature verifi
## text19.687                                                                                 they should find those
## text19.688                                                                          they should absolutely fin
## text20.80                                                                                               car prices w
## text19.792                                                                                              put it down
## text14.403                                                    so lets put aside the schoolyard taunts about whos
## text14.134                                                                                                      the
## text19.105                                                                                                    we wi
## text14.264                                                                                            premiums will
## text20.181                                                                                         we will finish t
## text14.308                                                            and if have to enforce this discipline by vet
## text14.419                                                                    but am absolutely confident we will s
## text20.141                                                              will you have the money to pay your medical
## text19.5                                                                          theyre not going to take it any
## text19.235                                                                                          and we pay the
## text14.358 and id urge democrats and republicans to pass bill that helps to correct some of these pr
## text13.61        and if you own business or farm you would find it harder and more expensive to get
## text20.49
## text19.71                                                                                       he fights and iil te
## text15.191                                                                                                so cezu t
```

```
# Are these call to actions?
```

Third, let's try to classify sentences using LDA.

```
dtm <- DocumentTermMatrix(corp, control = list(removeNumbers = TRUE,
                                               stemming = TRUE,
                                               removePunctuation = TRUE,
                                               stopwords = TRUE))
dtm <- dtm[apply(dtm,1,FUN=sum) != 0,]
# FindTopicsNumber(dtm, topics = seq(from = 2, to = 20, by = 1),
#   metrics = "CaoJuan2009", # other methods are "Arun2010" and "Deveaud2014"
#   method = "VEM") # should be 6 but too little
LDA <- LDA(dtm, k = 20, control = list(seed = 1234))
topics <- tidy(LDA, matrix = "beta") %>% # beta represents words in topic
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  mutate(terms = stringr::str_c(term, collapse=", ")) %>%
  group_by(terms, topic) %>%
  summarise(beta = mean(beta)) # get topics
corp_parse_lda <- tidy(LDA, matrix = "gamma") %>% # merge with NLP output
  group_by(document) %>%
  mutate(max_gamma = max(gamma)) %>%
  ungroup() %>%
  filter(max_gamma == gamma) %>%
  dplyr::left_join(parse, by = c("document" = "doc_id")) %>%
  dplyr::left_join(topics)
filter(corp_parse_lda, topic == 4) %>%
  arrange(-max_gamma) %>%
  slice_head(n = 1000) %>%
  group_by(document) %>%
```

```
  summarise(sentences = paste(token, collapse=" "))
```

```
## # A tibble: 33 x 2
##    document   sentences
##    <chr>      <chr>
##  1 text10.4   "The people 's trust has been my greatest source of strength to g~
##  2 text11.189 "He has shown instead utter contempt for the United Nations and f~
##  3 text11.246 "( APPLAUSE )   And the day he and his regime are removed from po~
##  4 text11.88  "One of them is found at the Healing Place Church in Baton Rouge ~
##  5 text12.146 "And meaningful institutional reforms must include measures to im~
##  6 text12.21  "That document commits this organization to work to " save succee~
##  7 text12.45  "We must defeat the terrorists on the battlefield , and we must a~
##  8 text14.258 "Still , this is a complex issue , and the longer it was debated ~
##  9 text14.364 "You 've trimmed some of this spending , you 've embraced some me~
## 10 text14.5   "It 's tempting to look back on these moments and assume that our~
## # i 23 more rows
```

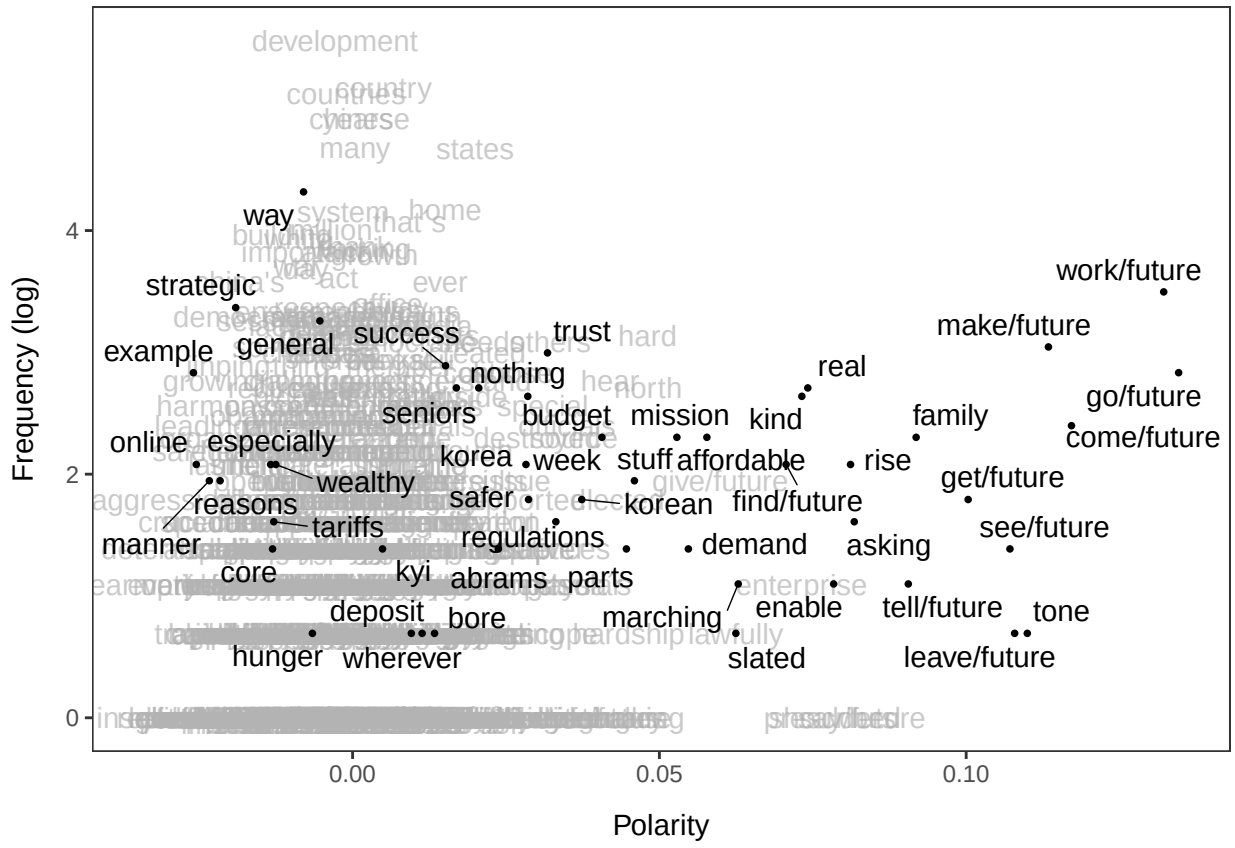
```
# Do these sentences belong together?
```

Urgency

Get temporality

Let's experiment with the LSX R package and employ Latent Semantic Scaling (LSS) to get both temporarily and necessity (sentiment in this case) in segmented text.

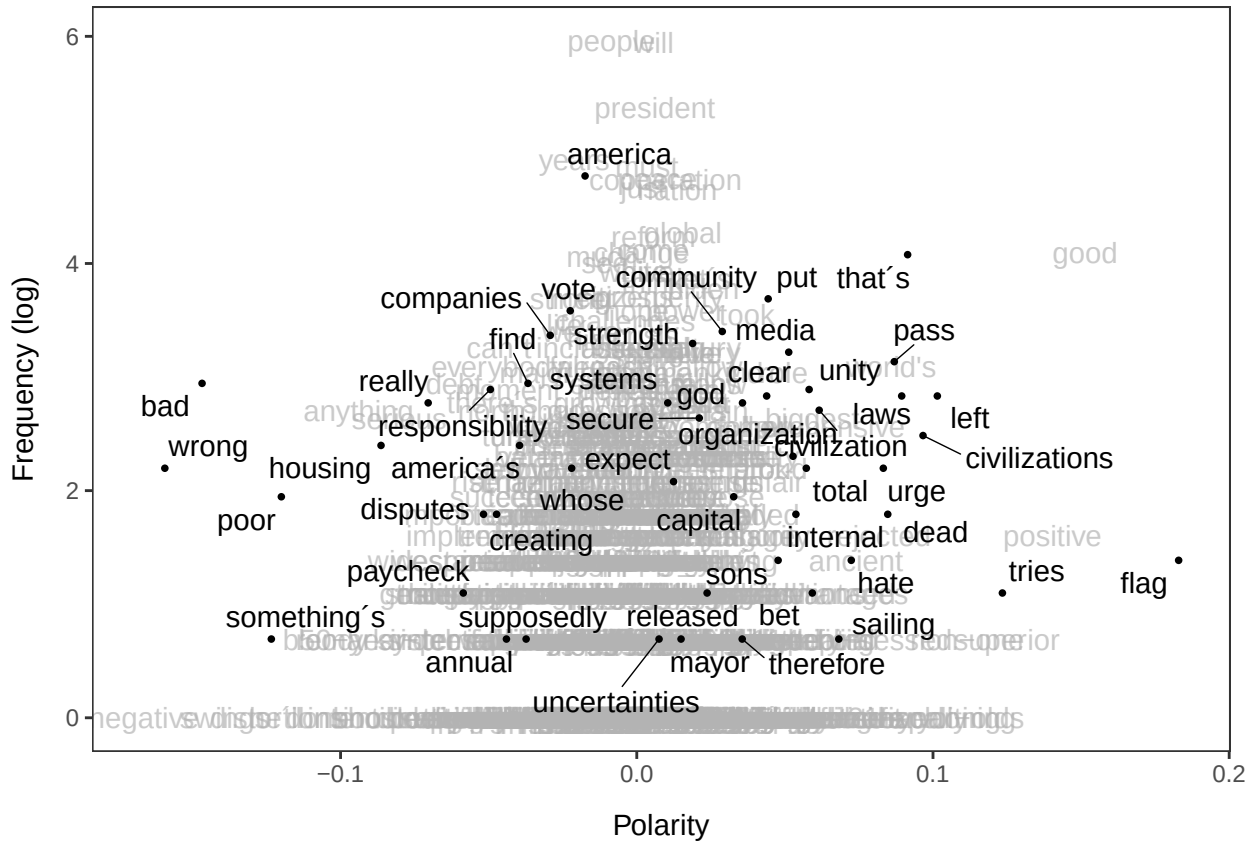
```
dict <- dictionary(file = "temporality.yml") # dictionary from https://journals.sagepub.com/doi/10.1177/
dfmt_future <- dfm_select(dfmt, dict$en$future["main"])
dfmt_future <- dfmt_future*as.integer(rowSums(dfm_select(dfmt, dict$en$future["aux"])) > 0)
colnames(dfmt_future) <- paste0(colnames(dfmt_future), "/future")
dfmt_perfect <- dfm_select(dfmt, dict$en$perfect["main"])
dfmt_perfect <- dfmt_perfect*as.integer(rowSums(dfm_select(dfmt, dict$en$perfect["aux"])) > 0)
colnames(dfmt_perfect) <- paste0(colnames(dfmt_perfect), "/past")
dfmt2 <- dfm_remove(dfmt, dict$en)
dfmt3 <- cbind(dfmt2, dfmt_future, dfmt_perfect)
seed <- c("*/future" = 1, "*/past" = -1)
lss <- textmodel_lss(dfmt3, seeds = seed, k = 300, cache = TRUE,
                     include_data = TRUE, group_data = TRUE, use_nan = TRUE)
textplot_terms(lss) # see terms
```




```
dat <- docvars(lss$data) # get data/vars
dat$lss <- predict(lss, min_n = 10) # predict
#knitr::kable(head(bootstrap_lss(lss, mode = "terms"), 10)) # evaluate terms
#knitr::kable(head(bootstrap_lss(lss, mode = "coef"), 10), digits = 3) # evaluate coefs
```

Get necessity (or sentiment)

```
# Sentiment could be useful since already pre-setup, no?
seed1 <- as.seedwords(data_dictionary_sentiment) # dictionary of words for training
lss1 <- textmodel_lss(dfmt, seeds = seed1, k = 300, cache = TRUE,
                      include_data = TRUE, group_data = TRUE) # model
textplot_terms(lss1) # see terms
```




```
dat1 <- docvars(lss1$data) # get data/vars
dat1$lss <- predict(lss1, min_n = 10) # predict
knitr::kable(head(bootstrap_lss(lss1, mode = "terms"), 10)) # evaluate terms
```

good	nice	positive	correct	superior	bad	poor	negative	wrong
good	nice	positive	correct	superior	bad	poor	negative	wrong
pretty	bea	refrain	blacklist	differ	cezu	shoulder	functioning	realism
answers	fiscally	exploiting	tweet	others	tin	mar	market	dogmas
necessarily	that's	abandoning	flag	civilization	seems	kets	triple-digit	discredited
cleansing	nurse	abrodiia	totally	age	main	forgiveness	swings	something's
motion	careful	factors	tries	birthplace	behavior	heavily	distortions	hostage
ratings	statement	mighty	radical	fate	rewarded	indebted	tolerated	principled
brother	oh	plays	left	uninterrupted	unwilling	transfer	defaults	choosing
region's	okay	advancing	media	unique	headline	developed	scandals	time
regime's	explain	favorable	put	world's	fire	heaviest	declines	old

```
knitr::kable(head(bootstrap_lss(lss1, mode = "coef"), 10), digits = 3) # evaluete coefs
```

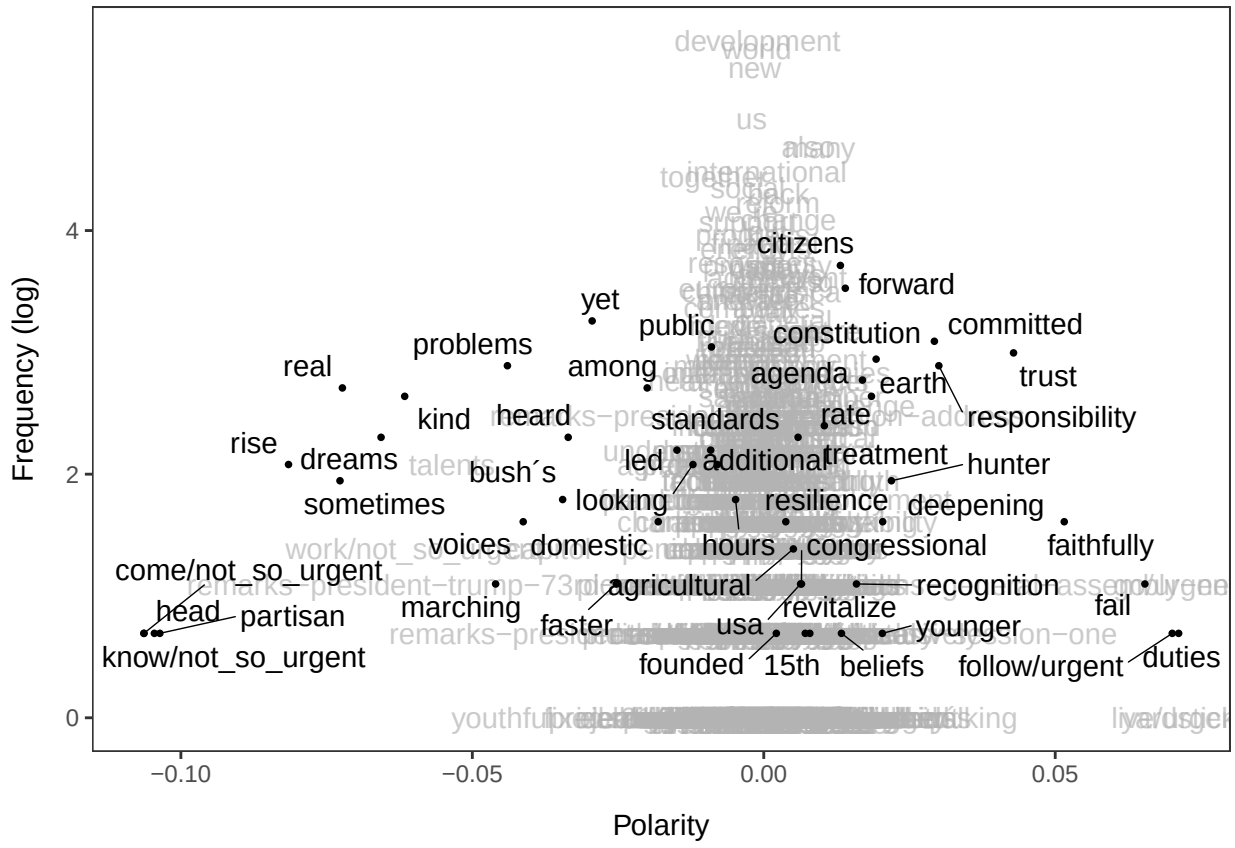
	good	nice	positive	correct	superior	bad	poor	negative	wrong
full	-0.009	-0.021	-0.025	0.018	0.016	-0.008	-0.027	-0.044	-0.007
text	-0.010	-0.011	0.009	0.003	0.037	-0.004	-0.002	0.045	0.000
hu's	-0.007	-0.020	-0.005	0.022	0.001	-0.009	-0.023	0.057	0.005
speech	-0.007	0.006	-0.004	-0.009	-0.006	-0.021	0.011	0.007	0.019
spdney	-0.012	0.000	-0.002	-0.024	0.022	-0.004	-0.020	-0.005	0.013

	good	nice	positive	correct	superior	bad	poor	negative	wrong
morning	0.045	-0.004	0.028	-0.107	0.002	0.077	-0.036	0.062	0.004
herald	-0.012	0.000	-0.002	-0.024	0.022	-0.004	-0.020	-0.005	0.013
national	0.005	-0.003	-0.067	0.018	-0.015	-0.011	-0.031	0.027	0.008
published	-0.003	-0.003	0.032	-0.018	0.083	-0.029	0.044	-0.014	0.007
years	0.004	0.005	0.001	0.000	-0.003	0.042	-0.033	-0.007	0.152

```
# Could we adapt this to be about intensity of call to action?
```

Perhaps I/we need to understand better what LSS does but it appears fast and efficient to classify words in terms of temporarily. Though, what we are interested in urgency, what about we create a dictionary with some of the words we coded?

```
dictu <- dictionary(file = "urgency.yml")
dfmt_urgent <- dfm_select(dfmt, dictu$en$urgent["main"])
dfmt_urgent <- dfmt_urgent*
  as.integer(rowSums(dfm_select(dfmt, dictu$en$urgent["aux"])) > 0)
dfmt_urgent <- dfmt_urgent*
  as.integer(rowSums(dfm_select(dfmt, dictu$en$urgent["urgent"])) > 0)
colnames(dfmt_urgent) <- paste0(colnames(dfmt_urgent), "/urgent")
dfmt_not_so_urgent <- dfm_select(dfmt, dictu$en$not_so_urgent["main"])
dfmt_not_so_urgent <- dfmt_not_so_urgent*
  as.integer(rowSums(dfm_select(dfmt, dictu$en$not_so_urgent["aux"])) > 0)
dfmt_not_so_urgent <- dfmt_not_so_urgent*
  as.integer(rowSums(dfm_select(dfmt, dictu$en$not_so_urgent["urgent"])) > 0)
colnames(dfmt_not_so_urgent) <- paste0(colnames(dfmt_not_so_urgent), "/not_so_urgent")
dfmt2u <- dfm_remove(dfmt, dictu$en)
dfmt3u <- cbind(dfmt2u, dfmt_urgent, dfmt_not_so_urgent)
seed <- c("*/urgent" = 1, "*/not_so_urgent" = -1)
lssu <- textmodel_lss(dfmt3u, seeds = seed, k = 300, cache = TRUE,
                      include_data = TRUE, group_data = TRUE, use_nan = TRUE)
textplot_terms(lssu) # see terms
```




```
datu <- docvars(lssu$data) # get data/vars
datu$lss <- predict(lssu, min_n = 10) # predict
#knitr::kable(head(bootstrap_lss(lssu, mode = "terms"), 10)) # evaluate terms
#knitr::kable(head(bootstrap_lss(lssu, mode = "coef"), 10), digits = 3) # coefs
```

While perhaps not a good dictionary, LSS could be useful to get/score urgency related token in text from a training set to construct a flexible dictionary every time that allows us to score a large number of words related to urgency.

Conclusions

- 1: How to store text?

Depends on what information we want to keep and how... I believe that combining an NLP output (by token) in a tidy text object (we can even create our own class for these types of objects) might be helpful. This approach allows us to easily change structure to larger text units (e.g. full text or sentences) or different types of text objects (e.g. corpus or document term matrix) when necessary while, at the same time, keeping as much metadata as possible at the token (word) level.

- 2: How to segment text into topics?

There is an opportunity to offer text tilling in R, though I am not sure how much work that would be. However, we need to be sure that this will get us to what we are looking for. That is, it is an automated approach that works relatively well but, at the end of the day, it is similar to other general text-topic classification algorithms (like LDA). Therefore, we can assume it is not generally precise unless parameters (e.g. number of topics) are tuned in relation to texts at hand. One alternative is to require users create their own dictionary (i.e. as a training set or simply word matches) or offer a broad text classification dictionary to

"organize" sub portions of texts into political topics (i.e. similar to `manypkgs::code_actions()`). Regardless of the choice, we need to think about how to capture our unit of analysis, "calls to action". That is, when someone is talking about a topic (or object of policy) we are interested in, when are they really expressing their preferences/choices/policies in relation to the topic instead of talking about past achievements (a.k.a. driving while looking at the rearview mirror)? Besides that, call to actions can be longer than one sentence and/or talk about multiple topics which complicates segmentation further.

- 3: How to score/rank urgency?

This requires we first define what urgency is, it's dimensions, and when it matters. In general, LSS might offer a flexible, scalable, and semi automated way to grasp with different urgency (words) in diverse corpus of texts. These words could them be used to score urgency in "calls to action".