Additional Resources
Arabic Text Analysis Workshop, Cairo University, March-April 2019
Rich Nielsen
rnielsen@mit.edu
nielsen.rich@gmail.com
Last updated: 5/15/2019

The following document responds to questions from the participants in the Cairo University workshop.

Facebook Data
After the American election in 2016, Facebook changed its API data access because of the Cambridge Analytica scandal. When I last checked in November 2018, there was no easy way for an individual researcher to use the Facebook API to get data. The best way for academic researchers to get data is by submitting a data request through Facebook's academic collaboration: Social Science One. https://socialscience.one/. I have not made any requests myself, so I cannot advise on how to be successful, but I am happy to help review a request you wish to make.

Twitter Data
A limited amount of data from Twitter is available through their API. You will need to sign up for a developer account (as if you were making an app). Here is a YouTube video that explains (though I have not watched it all): https://www.youtube.com/watch?v=-jvPDfz--l8
I have added a Twitter example at the end of the tutorial code.

R and R GUIs
R comes with a basic GUI for both Windows and Mac. I like the Windows version and use it myself. But I do not like the Mac version very much. Instead, I use RStudio, which is a free GUI for R that has additional functionality. Although the appearance is different, the results are the same. [UPDATE: I learned that the Mac version of R cannot generally display Arabic text in figures. The problem seems to be deep and I do not have a solution.]

Displaying Arabic correctly in RStudio
To display Arabic correctly in a script in RStudio, you must open it with the UTF-8 Encoding.

How to find examples of research using text analysis?
Scholars usually cite papers but do not cite software. Check Google Scholar citations to Grimmer and Stewart 2013 ("Text as data: The promise and pitfalls of automatic content analysis methods for political texts"):
https://scholar.google.com/scholar?cites=10738245560487335638

The STM website lists some papers that use the structural topic model software:
https://www.structuraltopicmodel.com/

Arabic Text Analysis in Python

I wrote a script to stem and transliterate Arabic in python. It is an older version of the arabicStemR package in R. The script is linked on my website: http://www.mit.edu/~rnielsen/helpful.htm and the code itself is on github: https://github.com/ChristopherLucas/txtorg/blob/65876dfb56a0b947df8a856bb0896a7e41e3f38c/textorganizer/nielsenstemmer.py

I sometimes work in both Python and R for text analysis. Python has excellent string manipulation commands via regex, and is faster at processing large numbers of documents. It is also much more sophisticated at web scraping. However, I find data analysis in python to be hard. There are now libraries for some statistical models, but not as many as R. My usual workflow is to do text collection with python and analysis in R. You can find any mix that works for you.

Information about arabicStemR

The manual for arabicStemR is here: https://cran.r-project.org/web/packages/arabicStemR/arabicStemR.pdf. It is somewhat helpful, but not exhaustive.

Transliteration

Arabic letter	Transliterated
ا	a
ى	A
ب	b
ت	t
ث	U
ج	j
ح	7
خ	K
د	d
ذ	i
ر	r
ز	z
س	s
ش	W
ص	S
ض	D
ط	T
ظ	Z
ع	3
غ	G
ف	f

ق	Q
ك	K
ل	l
م	m
ن	n
ه	h
و	w
ي	y
أ	a
إ	a
ؤ	o
ئ	5
ء	q
آ	a
ة	0
پ	p
ھ	h
ک	k
ٹ	t
ں	n
ے	y
ی	y
اً	a
لا	l
ھ	h
ۓ	y
ێ	y
ڭ	k
ڪ	k

Stop words removed by default in stem() and removeStopWords()

Prepositions

"في", "فيه", "فيها", "فيهم", "على",
"عليك", "عليكم", "علينا", "عليه", "عليها",
"عليهم", "علي", "به", "بها", "بهم", "بهذا",
"بذلك", "بك", "بكم", "بكل", "بما", "بمن",
"بنا", "له", "لها", "لهم", "مع", "معه",
"معها", "معهم", "عن", "عنا", "عنه", "عنها",
"عنهم", "تحت", "حتى", "فوق", "فوقَ",
"بجانب", "أمام", "أمامَ", "امام", "خارج",
"بالخارج", "حولَ", "حول", "رغم", "بالرغم",

"رغمَ", "منذ", "منذُ", "من", "خلال",
"خلالَ", "حول", "حولَ", "قبل", "قبلَ",
"وفقا", "إلى", "الوراءَ", "وراء",
"بينَ", "بين", "بينهم", "بينهما", "بينكم",
"بينما", "بدون", "لكن", "باتجاه", "أقل",
"اقل", "اكثر"

Pronouns

"هذا", "هذه", "ذلك", "تلك", "هؤلَاء",
"هؤلاء", "اولائك", "هذان", "هذينهتان",
"هتينأنا", "انا", "أنت", "هما", "أنتَ",
"انت", "أنت", "أنتِ", "انتهو", "هوَ",
"هو", "هي", "هيَ", "نحن", "أنتم", "انتم",
"أنتم", "انتم", "هُم", "هم", "لهم", "منهم",
"وهم", "التي", "الذي", "اللذان", "اللذين",
"اللتان", "اللتين"

Particles and connectors

"ان", "وان", "إن", "إنه", "إنها",
"إنهم", "إنهما", "إني", "وإن", "وأن",
"ان", "انه", "انها", "انهم", "انهما",
"اني", "أنك", "إنك", "انك", "أنكم", "إنكم",
"انكم", "اننا", "وان", "وان", "أن", "ان",
"ألا", "بأن", "ان", "الا", "بان", "بانهم",
"أنه", "أنها", "أنهم", "أنهما", "انه",
"اذ", "انها", "انهم", "انهما", "أذ",
"اذا", "إذ", "إذا", "وإذ", "وإذا", "اذ",
"اذ", "اذا", "اذ", "اذا", "فاذا", "ماذا",
"واذ", "واذا", "لولا", "لو", "ولوسوف",
"لن", "ما", "لم", "ولم", "أما", "اما",
"لا", "ولا", "إلا", "الا", "أم", "أو",
"ام", "او", "بل", "قد", "وقد", "لقد", "أنما",
"إنما", "بل", "انما", "انما", "و",
"بما", "كما", "لما", "لأن", "لان",
"لي", "لى", "لهذأ", "لذأ", "لأنه", "لأنها",
"لأنهم", "لان", "لانه", "لانها", "لانهم",
"ثم", "أيضا", "ايضا", "كذلك", "قبل",
"بعد", "لكن", "ولكن", "لكنه", "لكنها",
"لكنهم", "فقط", "رغم", "بالرغم", "بفضل",
"حيث", "بحيث", "لكي", "هنا", "هناك",
"بسبب", "ذات", "ذو", "ذي", "ذى", "وه",
"يا", "انما", "فهذا", "فهو", "فما", "فمن",
"فيما", "فهل", "وهل", "فهؤلاء", "كذا",
"لذلك", "لماذا", "لمن", "لنا", "منا",
"منك", "منكم", "منهما", "منهما", "لك",

"ولو", "مما", "وما", "ومن", "عند", "عندهم",
"اذن", "عنك", "عنهما", "عندنا", "عندما",
"فه", "فهم", "فانهم", "فانا", "الذي",
"فيك", "فلما", "فلم", "لكم", "لكل", "فكل",
"لهذا", "فيكم"

Prefix removal

Only one prefix is removed. They are evaluated in this order – after the first prefix match is found and removed, the stemmer moves to the next word.

"ال" if >= 4 characters
"وال" if >= 5 characters
"بال" if >= 5 characters
"كال" if >= 5 characters
"فال" if >= 5 characters
"لل" if >= 5 characters
"و" if >= 4 characters

Prefix removal

Only one prefix is removed. They are evaluated in this order – after the first prefix match is found and removed, the stemmer moves to the next word.

"ها" if >= 4 characters
"ان" if >= 4 characters
"ات" if >= 4 characters
"ون" if >= 4 characters
"ين" if >= 4 characters
"يه" if >= 4 characters
"ية" if >= 4 characters
"ه" if >= 3 characters
"ة" if >= 3 characters
"ي" if >= 3 characters

Is there a way to turn off parts of the stemmer without programming?
Sort of. In the commands “removePrefixes()” and “removeSuffixes()”, you can specify how long a word must be in order to remove the stem. If you set this number very high for a specific prefix or suffix (i.e, Inf), it will not remove that prefix or suffix. However, this is **inside** the stem() function. You would have to combine your own custom stemmer from the internal parts, which does require programming. I have an example with code here: http://www.mit.edu/~rnielsen/r%20stemmer%20example_website.R

How can I see what the stemmer is doing?

In the code, I created an object called "stemListHolder." This object is a list where each element is a vector. In each vector, each element is a stem, and the name of the element is the original word. We can use this to see exactly what the stemming did to any word in the corpus.

Assuming the stemListHolder object is in the memory, the code below makes the calculations (note, I have added this code to the tutorial).

```
## How many words were stemmed in some way?
table(unlist(lapply(stemListHolder, function(x){x!=names(x)})))

## How many unique words are there without stemming?
length(unique(names(unlist(stemListHolder))))
## How many unique words are there with stemming?
length(unique(unlist(stemListHolder)))

## Get the suffixes and prefixes for the daeyat
suffixesAndPrefixesHolder <- c()
for(j in 1:length(stemListHolder )){
  print(paste(j,"of",length(stemListHolder )))
  suffixesAndPrefixes <- c()
  if(length(stemListHolder[[j]])==0){next}
  for(i in 1:length(stemListHolder[[j]])){
    stemmedunit <- stemListHolder[[j]][i]
    suffixesAndPrefixes <-
c(suffixesAndPrefixes,strsplit(names(stemmedunit),stemmedunit)[[1]])
  }
  suffixesAndPrefixesHolder[[j]] <- suffixesAndPrefixes[-
which(suffixesAndPrefixes=="")]
}
## Table the result so we have counts for each prefix and suffixe
suffixesAndPrefixesTab <- sort(table(unlist(suffixesAndPrefixesHolder)))
## List them one by one (because the Arabic names get reversed
## if we print the table all together)
for(i in 1:length(suffixesAndPrefixesTab)){
  print(suffixesAndPrefixesTab[i])
}
```

Add bigrams to a dtm

(note, I have added this code to the tutorial).

```
## Add bigrams to the dtm

mybigram <- "nad zmalk"

## To count a single bigram, you could do this and add the column to the DTM
unlist(lapply(str_extract_all(dat$text,mybigram),length))

## To add a single bigram to all documents, you could do this and paste it to the
## documents before making a DTM.
unlist(lapply(str_extract_all(dat$text,mybigram),paste,collapse=" "))

dat$bigramText <- paste(dat$text,
unlist(lapply(str_extract_all(dat$text,mybigram),paste,collapse=" ")))
```

```
## if we wanted to learn ALL of the bigrams around a particular stem
## we can do something more complicated

myunigram <- "nad"

textBigrams <- rep(NA, length(dat$text))
for(i in 1:length(dat$text)){
  mydoc <- dat$text[i]
  if(length(grep(myunigram,mydoc))>0){
    bigrams1 <- str_extract_all(mydoc,paste0("[0-9a-zA-z]+ [0-9a-zA-
z]*",myunigram,"[0-9a-zA-z]*"))[[1]]
    bigrams2 <- str_extract_all(mydoc,paste0("[0-9a-zA-z]*",myunigram,"[0-9a-zA-z]*
[0-9a-zA-z]+"))[[1]]
    bigrams1 <- paste0(sapply(bigrams1,function(x){strsplit(x," ")[[1]][1]}),"-
",myunigram)
    bigrams2 <- paste0(myunigram,"-",sapply(bigrams2,function(x){strsplit(x,"
")[[1]][2]}))
    textBigrams[i] <- paste(paste(bigrams1, collapse=" "), paste(bigrams2, collapse="
"))
  } else {
    textBigrams[i] <- ""
  }
}
## We can look at what bigrams there are
sort(table(strsplit(paste(na.omit(textBigrams),collapse=" "),")[[1]]))
## We could append these to the original documents before making
## the dtm
dat$text2 <- paste(dat$text,textBigrams)
## Then we would make the dtm with "dat$text2"
```

What is next after this workshop?

I recommend trying to do a project. Experience is the best teacher.

There are many resources online for learning more:
- A set of talks given at a "Text as Data" workshop in 2010.
https://toolsfortext.wordpress.com/readings-and-software/
- Justin Grimmer's "Text As Data" class materials are available here:
https://www.justingrimmer.org/teaching.html
- See day 3 of this short course: https://compsocialscience.github.io/summer-institute/2018/teaching-learning-materials
- There are a number of books on text analysis in R:
https://www.tidytextmining.com/
- There are papers that explain text analysis in R:
https://kenbenoit.net/pdfs/text_analysis_in_R.pdf
- There are websites with examples for various R resources:
https://quanteda.io/

But really, I recommend diving into a project and learning to search on the web for materials to learn the skills you need for that project.