N-grams : Tidytext Vs tm

This notebook demonstrates a difference I found in computing n-grams from the 'tidytext' package versus the 'tm' package, when the corpus contains multiple documents/lines.

Create a little example corpus, where EOS signifies end of sentence:

```
rm(list=ls())
corpus <- c("buy the book EOS","buy the book EOS")
corpus
```

```
## [1] "buy the book EOS" "buy the book EOS"
```

Tokenize w/ tidytext:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidytext)

df <- data_frame(corpus)
uni <- unnest_tokens(df,ngram,corpus,token="words")
bi <- unnest_tokens(df,ngram,corpus,token="ngrams",n=2)
tri <- unnest_tokens(df,ngram,corpus,token="ngrams",n=3)
```

Tokenize w/ tm:

```
library(tm)
```

```
## Loading required package: NLP
```

```
library(RWeka)
mysource <- VectorSource(corpus)
corp <- VCorpus(mysource)

# find bigrams
bigramTokenizer <-  function(x) unlist(lapply(ngrams(words(x), 2), paste, collapse = " "), use.names = F
bi_tm <- TermDocumentMatrix(corp, control = list(tokenize = bigramTokenizer))

# find trigrams
trigramTokenizer <-  function(x) unlist(lapply(ngrams(words(x), 3), paste, collapse = " "), use.names =
tri_tm <- TermDocumentMatrix(corp, control = list(tokenize = trigramTokenizer))
```

We see that there are 4 unique bigrams from the tidytext method:

```
bi_ct<-bi %>% count(ngram,sort=TRUE)
bi_ct
```

```
## # A tibble: 4 × 2
```

```
##      ngram     n
##      <chr> <int>
## 1 book eos     2
## 2  buy the     2
## 3 the book     2
## 4  eos buy     1
```

While there are only 3 unique bigrams from the tm package:

```
data_frame(rownames(as.matrix(bi_tm)))
```

```
## # A tibble: 3 × 1
##   `rownames(as.matrix(bi_tm))`
##                          <chr>
## 1                     book eos
## 2                      buy the
## 3                     the book
```

What I think happened is that the tm package computed n-grams for each 'document' or line of the corpus, while the tidytext package computed n-grams as if all the documents were one long line (the bigram 'eos buy' spans the 2 lines of the corpus). If we were to remove the titdytext n-grams where EOS is at the beginnging or middle of a sentence, the results would match.

```
sessionInfo()
```

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X Yosemite 10.10.5
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] RWeka_0.4-29   tm_0.6-2       NLP_0.1-9      tidytext_0.1.2
## [5] dplyr_0.5.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.7       knitr_1.15        magrittr_1.5
##  [4] RWekajars_3.9.0-1 mnormt_1.5-5      lattice_0.20-34
##  [7] R6_2.2.0          stringr_1.1.0     plyr_1.8.4
## [10] tools_3.3.2       parallel_3.3.2    grid_3.3.2
## [13] broom_0.4.1       nlme_3.1-128      psych_1.6.9
## [16] DBI_0.5-1         janeaustenr_0.1.4 htmltools_0.3.5
## [19] lazyeval_0.2.0    yaml_2.1.13       assertthat_0.1
## [22] digest_0.6.10     tibble_1.2        Matrix_1.2-7.1
## [25] rJava_0.9-8       purrr_0.2.2       tidyr_0.6.0
## [28] reshape2_1.4.2    tokenizers_0.1.4  SnowballC_0.5.1
## [31] slam_0.1-38       evaluate_0.10     rmarkdown_1.1
## [34] stringi_1.1.2     foreign_0.8-67
```