Appendix: Technical Walkthrough, ECB Text Sentiment Analysis

# Executive summary

In the report below, we performed text sentiment analysis on the ECB press conference speeches. Four different sentiment dictionaries were used for comparison, with words identified as positive contributing a positive score and negative words contributing a negative score. The analysis is done on unigrams (single word) basis, with adjustment for negation words through bigrams. The sentiment indicator is then built up from the summation of the scores and its change measured across time.

library(dplyr)  
library(tidytext)  
library(lubridate)  
library(ggplot2)  
library(tidyverse)  
library(GGally)  
source("text\_analytics.R") # Stores many of the functions used below

# Methodology

### Read in ECB speeches as scraped

The transcript of ECB press conference speeches were separately scraped from the ECB website, and we start by reading them into dataframes. We filter out text which are questions asked by the press audience

df <- read.csv("data/ecb\_speeches.csv", stringsAsFactors = FALSE) %>%  
 mutate(date = ymd(date), type = as.factor(type), speaker = as.factor(speaker))  
  
# Extract press speech and answers  
df\_ecb <- df %>%  
 filter(type %in% c("speech", "answer")) %>%  
 group\_by(date) %>%  
 summarise(text = paste0(text, collapse = "."))  
  
str(df\_ecb)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 217 obs. of 2 variables:  
## $ date: Date, format: "1998-06-09" "1998-07-08" ...  
## $ text: chr "I may recall that the Heads of State or Government appointed six members to the Executive Board of the ECB, inc"| \_\_truncated\_\_ "The Governing Council first assessed current economic developments in the euro area. The general picture is one"| \_\_truncated\_\_ "It is my intention to continue this practice for the remaining meetings of this year. In this regard, the Gover"| \_\_truncated\_\_ "Ladies and gentlemen, in line with our stated intention, the Vice-President and I are here today to report on t"| \_\_truncated\_\_ ...

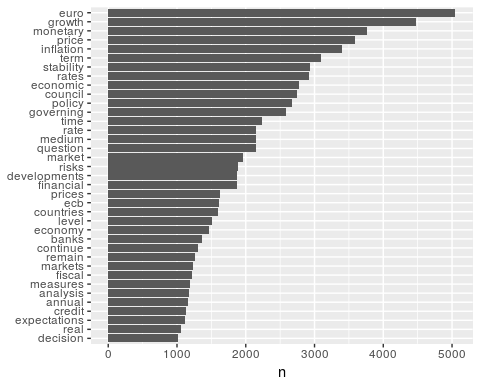
## Preprocess: Tokenising words and removing stop words

The next step is to convert paragraphs of text to individual words, in a process known as tokenising. We start by analysing single words, also referred to as unigrams.

In addition, we remove stop words which are words used very often which convey no meaning such as "is", "the" etc. We added our own custom stop words to remove years.

Performing a simple count analysis on the unigrams, we observe that words like "euro", "growth" are used very often in the speeches.

#Remove years as stop words  
custom\_stop\_words = bind\_rows(data\_frame(word = c(as.character(1980:2030)),  
 lexicon = c("custom")),  
 stop\_words)  
  
ecb\_unigrams <- df\_ecb %>%  
 make\_ngrams(custom\_stop\_words = custom\_stop\_words)   
  
ecb\_unigrams %>%  
 count\_ngrams %>%  
 filter(n > 1000) %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(word, n)) +  
 geom\_col() +  
 xlab(NULL) +  
 coord\_flip()



## Single word sentiment analysis

### List of top words and their associated sentiments

After obtaining the count of the words, we now need to find their associated sentiments. We use four dictionaries to measure sentiment, however these dictionaries are not optimised for central bank or macroeconomic analysis and is sub optimal.

|  |  |
| --- | --- |
| Dictionary | Description |
| AFINN | Scale of -5 (very negative) to +5 (very positive) |
| Loughran | For financial applications, various types of sentiments |
| NRC | Crowd sourced emotion-based |
| Bing | Labels as positive or negative |

From an extract of the top words with associated sentiments, we see that the sentiments associated with each dictionary is quite varied.

# Compare how words categorization differs across sets  
ecb\_unigrams %>%  
 count\_ngrams %>%  
 add\_sentiments(c("afinn", "loughran", "nrc", "bing")) %>%  
 compare\_words\_sentiment

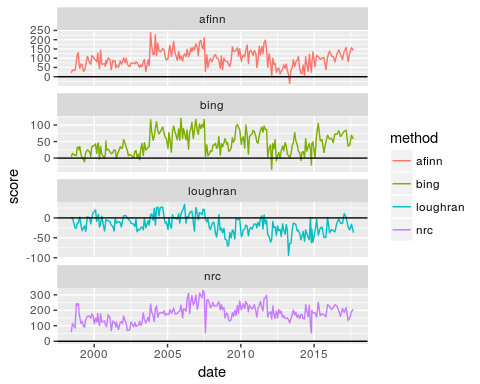
## # A tibble: 3,208 x 6  
## # Groups: word, n [3,208]  
## word n afinn bing loughran nrc  
## <chr> <int> <int> <chr> <chr> <chr>  
## 1 growth 4483 2 <NA> <NA> positive  
## 2 monetary 3760 NA <NA> <NA> anticipation|positive  
## 3 inflation 3396 NA <NA> <NA> fear|negative  
## 4 stability 2930 NA positive positive <NA>  
## 5 council 2738 NA <NA> <NA> anticipation|positive|trust  
## 6 policy 2676 NA <NA> <NA> trust  
## 7 time 2243 NA <NA> <NA> anticipation  
## 8 question 2142 NA <NA> negative positive  
## 9 risks 1894 -2 negative uncertainty <NA>  
## 10 level 1511 NA <NA> <NA> positive|trust  
## # ... with 3,198 more rows

### Calculating a sentiment score

We then calculate a sentiment score by adding to the score when a positive word appears and subtracting when a negative word appears. In the case of the AFINN dictionary, we multiple the word count n by the sentiment score before summing them together. Here we observe that each indicator is fairly diferent, suggesting that the dictionary has significant impact on your final results.

Using zero as a baseline, we observe that most of the dictionaries tend to give a positive overall sentiment to the minutes, but Loughran views the speeches negatively in most cases

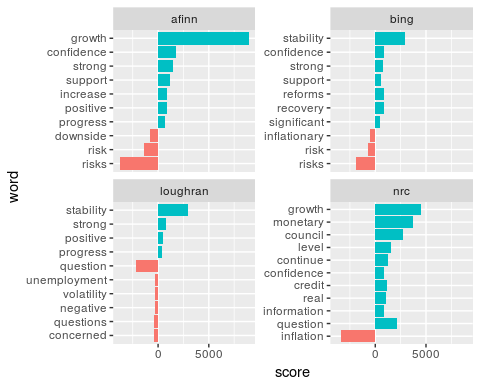
ecb\_scores <- ecb\_unigrams %>%  
 add\_sentiments(all = TRUE) %>%  
 group\_by(date) %>%  
 calc\_sentiment\_score(wt = "n")  
  
g <- ecb\_scores %>%   
 ggplot(aes(date, score, colour = method))   
g + geom\_line() +   
 geom\_hline(yintercept = 0) +  
 facet\_wrap(~method, ncol = 1, scales = "free\_y")



### High scoring words from the dictionary

To account for the differences, we extracted the top contributors to scores both negative and positive across the dictionaries, and see that they are indeed picking out quite different words

ecb\_unigrams %>%  
 add\_sentiments(all = TRUE) %>%  
 group\_by(word) %>%  
 calc\_sentiment\_score(wt = "n") %>%  
 group\_by(method) %>%  
 top\_n(10, abs(score)) %>%  
 ungroup %>%  
 mutate(word = reorder(word, score)) %>%   
 ggplot(aes(word, score, fill=factor(score > 0))) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~method, scales = "free\_y") +  
 coord\_flip()



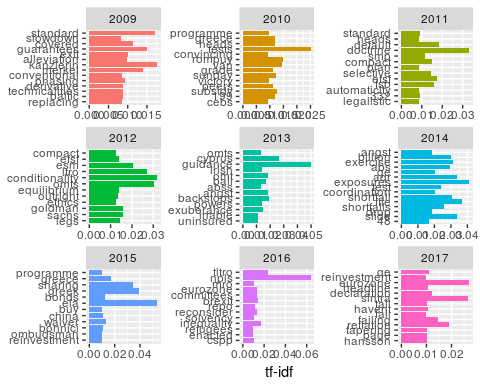
### Calculate ranking for tf-idf

It is arguable that a pure word count is an accurate reflection of the sentiments in a set of minutes. As words like growth or inflation may be used very often in any ECB speech. Hence we may wish to adjust for the frequency of these terms appearing in the documents.

We use term frequency - inverse document frequency (tf-idt) as a weight. tf-idf measures how unique a particular word is in the other documents. If it is relatively unique, it is given a higher weight, while if it appears frequently in every single document, it would be given a smaller weight.

Below we see words with high weights due to the uniqueness of their appearance for each of the years

unigrams\_wt <- ecb\_unigrams %>%  
 group\_by(date) %>%  
 count\_ngrams %>%  
 bind\_tf\_idf(word, date, n) %>%  
 ungroup  
  
unigrams\_wt %>%   
 ungroup %>%  
 filter(year(date) > 2008) %>%  
 group\_by(year = as.factor(year(date))) %>%   
 top\_n(15, tf\_idf) %>%   
 ungroup %>%  
 ggplot(aes(factor(word, levels = rev(unique(word))), tf\_idf, fill = year)) +  
 geom\_col(show.legend = FALSE) +  
 labs(x = NULL, y = "tf-idf") +  
 facet\_wrap(~year, ncol = 3, scales = "free") +  
 coord\_flip()



### Calculation of weighted scores

With the weights, we do a multiplication of the weights with the sentiment to give us the weighted score. To compare with the unweighted scores, we standardize both the weighted and unweighted scores to compare them on the same scale. We see that directionally, both looks quite similar.

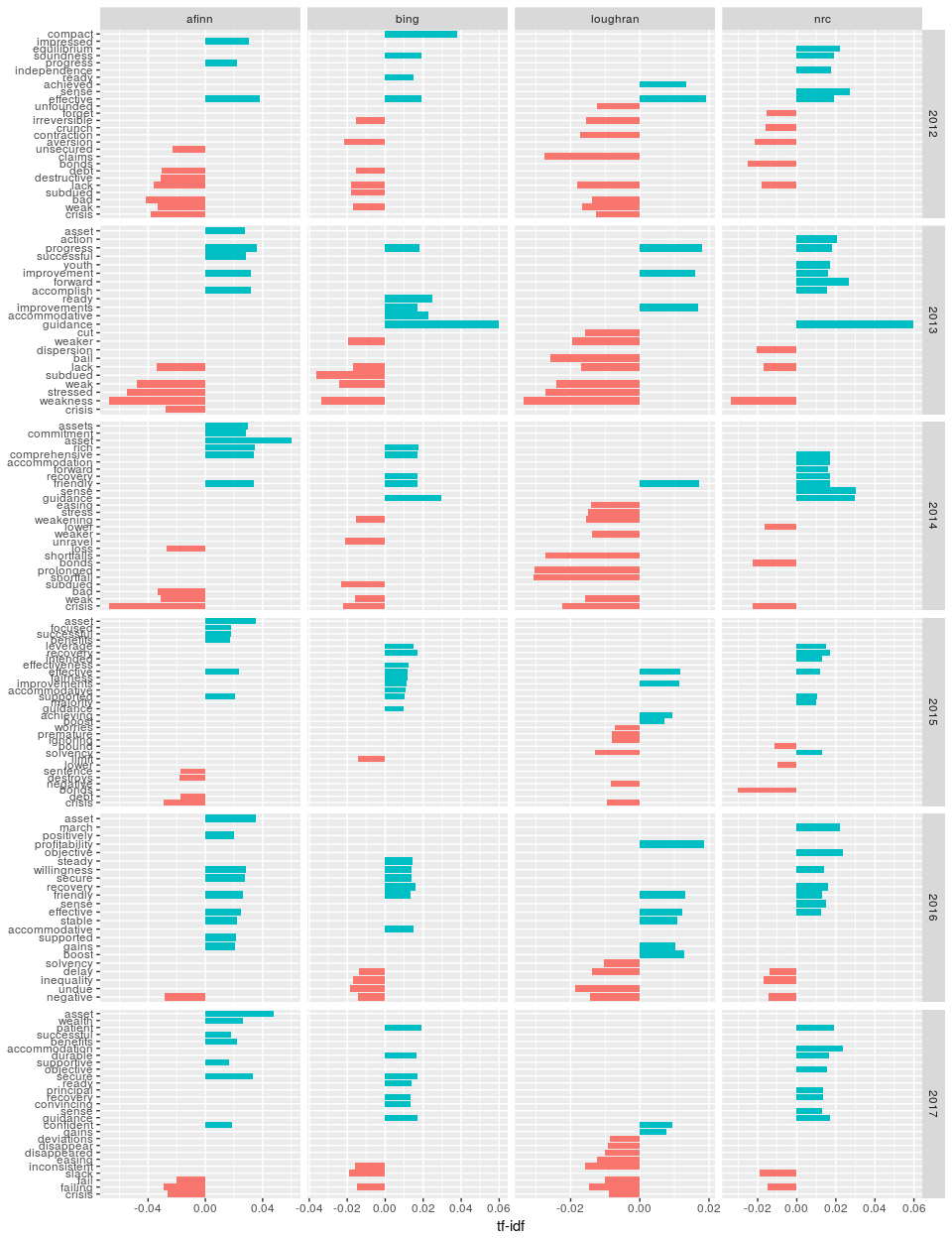
weighted\_scores <- unigrams\_wt %>%  
 add\_sentiments(all = TRUE) %>%  
 group\_by(date, method) %>%  
 calc\_sentiment\_score(wt = "tf\_idf")   
  
combined\_scores <- bind\_rows(ecb\_scores %>% mutate(weight = "unwt"),  
 weighted\_scores %>% mutate(weight = "wt"))  
  
# Centre all scores and plot weight and unweighted on the same chart  
combined\_scores %>%  
 group\_by(method, weight) %>%  
 mutate(score = scale(score)) %>%  
 ggplot(aes(date, score, colour = weight)) +  
 geom\_line() +  
 facet\_wrap(~method, ncol = 1, scales = "free\_y") +  
 geom\_hline(yintercept = 0, lwd=0.3, lty = 5) +  
 ggtitle("Standardized sentiment scores")



### Top contributors to weighted score

Here are the list of top contributors to weighted score for each of the recent years since 2011

unigrams\_wt %>%  
 add\_sentiments(all = TRUE) %>%  
 group\_by(date, word) %>%  
 calc\_sentiment\_score(wt = "tf\_idf") %>%  
 group\_by(year = year(date), word, method) %>%  
 summarise(score = sum(score)) %>%  
 group\_by(year, method) %>%  
 filter(year > 2011) %>%  
 top\_n(10, abs(score)) %>%  
 ungroup %>%  
 ggplot(aes(factor(word, levels = unique(word[order(score)])), score, fill = factor(sign(score)))) +  
 geom\_col(show.legend = FALSE) +  
 labs(x = NULL, y = "tf-idf") +  
 facet\_grid(year~method, scales = "free") +  
 coord\_flip()

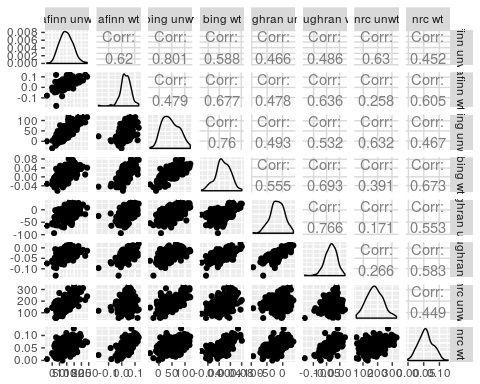


## Diagnostics of using different dictionaries

### Comparison of weighted scores across the different dictionaries

Correlations are quite weak across dictionaries as well as when considering weighted or unweighted

combined\_scores %>%  
 mutate(method = paste(method, weight)) %>%  
 select(-negative, -positive, -weight) %>%  
 spread(method, score) %>%  
 select(-date) %>%  
 ggpairs



## Adjusting for bigrams

### Removing negation words

Unigram analysis may not be sufficient, as there are words which are used with opposite intentions, such as "negative growth" or "no money". Hence we extend the analysis to bigrams, which are essentially words paired together with their adjacent words.

From an algorithm perspective, you may have noticed that to do such bigram analysis, we cannot remove stop words when tokenizing, as words like "not", "no" are considered stop words.

We identify common negation words such as "not", "no" etc.

From the identified word list, we do see that in the unigram analysis, we would have misclassified the sentiment of these words

bigrams\_separated <- df\_ecb %>%  
 make\_ngrams(2, remove\_stop\_words = FALSE)  
  
negation\_words <- c("not", "no", "never", "without", "negative", "weak")  
  
negated\_words <- bigrams\_separated %>%  
 filter(word1 %in% negation\_words) %>%  
 ungroup()  
  
negated\_words %>%   
 rename(word = word2) %>%   
 add\_sentiments %>%   
 filter(sentiment %in% c("positive", "negative"))

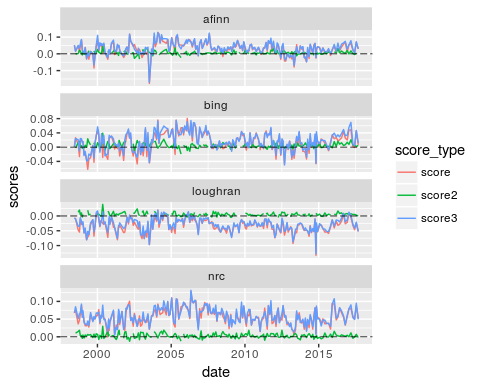
## # A tibble: 2,136 x 6  
## date word1 word n sentiment method  
## <date> <chr> <chr> <int> <chr> <chr>  
## 1 2008-10-02 not disappeared 5 negative loughran  
## 2 2003-11-06 not mistaken 3 negative loughran  
## 3 2006-02-02 not easy 3 positive loughran  
## 4 2007-03-08 not abrupt 3 negative loughran  
## 5 2008-01-10 not tolerate 3 negative loughran  
## 6 2008-11-06 not disappeared 3 negative loughran  
## 7 2014-08-07 no doubt 3 negative loughran  
## 8 2016-06-02 without undue 3 negative loughran  
## 9 1999-09-09 no problem 2 negative loughran  
## 10 2000-05-11 no danger 2 negative loughran  
## # ... with 2,126 more rows

### Adjusting for negated words

To adjust the negated words, we have to first remove its original impact, then add the new impact, which is equivalent to adding twice its new established impact

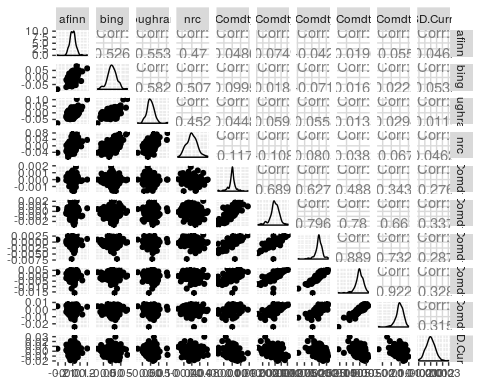
##################  
# adjust weighted scores  
##################  
  
# Calculate negative scores from negated words  
negative\_scores\_by\_word <- negated\_words %>%   
 rename(n2 = n, word = word2) %>%  
 inner\_join(unigrams\_wt, by = c("date", "word")) %>%  
 mutate(tf\_idf2 = (n2 \* 2 \* -1)/n \* tf\_idf) %>% # We subtract twice to adjust for the "wrong" impact on the raw score  
 add\_sentiments(all = TRUE) %>%  
 group\_by(date, word, word1) %>%  
 calc\_sentiment\_score(wt = "tf\_idf2")   
  
# Total negative scores by date  
negative\_scores <- negative\_scores\_by\_word %>%  
 group\_by(date, method) %>%  
 summarise(score = sum(score))  
  
# Join negative scores to original weighted scores, and recalculated combined scores  
bigram\_adjusted\_scores <- negative\_scores %>%  
 # select(-negative, -positive) %>%  
 rename(score2 = score) %>%  
 right\_join(weighted\_scores, by = c("date", "method")) %>%  
 mutate(score3 = ifelse(is.na(score2), score, score + score2))  
  
# Plot combined weighted scores scores  
bigram\_adjusted\_scores %>% select(date, method, score, score2, score3) %>%  
 gather(score\_type, scores, -date, -method) %>%  
 ggplot(aes(date, scores, colour = score\_type)) +  
 geom\_line() +  
 facet\_wrap(~method, ncol=1, scales = "free\_y") +  
 geom\_hline(yintercept = 0, lwd=0.3, lty = 5)

## Warning: Removed 1 rows containing missing values (geom\_path).

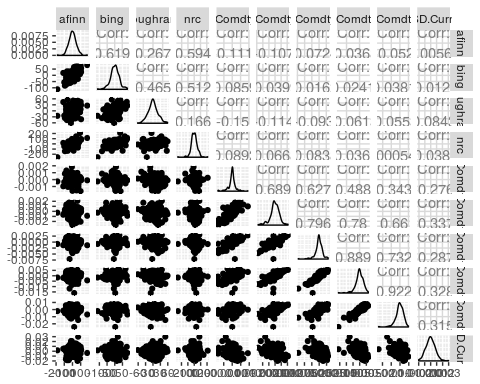


# Plot words with negated contributions  
# negative\_scores\_by\_word %>%  
# group\_by(word, word1) %>%  
# summarise(score = sum(score)) %>%  
# top\_n(20, desc(abs(score))) %>%  
# mutate(full\_word = paste(word1, word)) %>%  
# ggplot(aes(score, full\_word, fill = score > 0)) %>%  
# geom\_col() %>%  
# coord\_flip()  
   
   
  
####################  
# adjust unweighted scores  
####################  
  
#negated\_words %>%   
  
# weighted\_scores  
# weighted\_scores %>% inner\_join(negative\_scores, by = c("date", "method"))  
#   
# negated\_words %>%  
# rename(word = word2) %>%  
# add\_sentiments(all = TRUE) %>%  
# group\_by(date, word1, word) %>%  
# calc\_sentiment\_score(wt = "") %>%  
# mutate(contribution = n \* score) %>%  
# group\_by(word1) %>%  
# top\_n(10, desc(abs(contribution))) %>%  
# arrange(contribution) %>%  
# ungroup %>%  
# mutate(word2 = reorder(word2, contribution)) %>%  
# ggplot(aes(word2, n \* score, fill = n \* score > 0)) +  
# geom\_col(show.legend = FALSE) +  
# facet\_wrap(~word1, ncol = 2, scales = "free\_y") +  
# xlab("Words preceded by \"not\"") +  
# ylab("Sentiment score \* number of occurrences") +  
# coord\_flip() +  
# scale\_color\_manual(labels = c("original", "negatives", "adjusted"), values = c("red", "green", "blue"))

bonds\_raw <- read.csv("data/asset\_data.csv")  
  
bonds\_data <- bonds\_raw %>%  
 mutate\_at(.vars = vars(-Date), .funs = funs(ret = lead(.,0)/lag(., 1)-1)) %>%  
 mutate(date = mdy(Date)) %>%  
 select(date, ends\_with("ret"))  
  
all\_data <- bigram\_adjusted\_scores %>%  
 group\_by(method) %>%  
 mutate(adj\_score = score3) %>%  
 select(date, method, adj\_score) %>%  
 mutate(adj\_score = c(NA,diff(adj\_score))) %>% # Measure changes in sentiment between speeches  
 spread(method, -date) %>%  
 left\_join(bonds\_data, by = c("date")) %>%  
 na.omit  
   
all\_data %>%  
 ungroup %>%  
 select(-date) %>%  
 ggpairs



ecb\_scores %>%  
 group\_by(method) %>%  
 select(date, method, score) %>%  
 mutate(score = score - lag(score, 1)) %>%  
 spread(method, -date) %>%  
 left\_join(bonds\_data, by = c("date")) %>%  
 na.omit %>%  
 select(-date) %>%  
 ggpairs

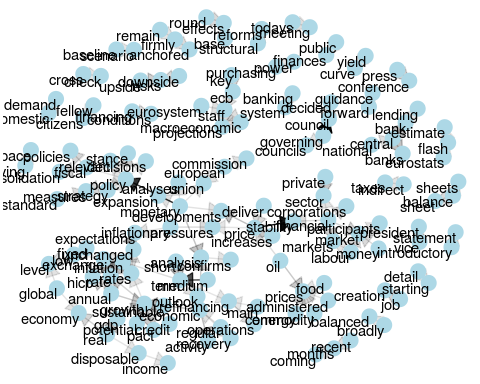


# Other text analytics tools

## Visualizing network of bigrams

Other functionalities for future work include chaining of bigrams, which allow for deeper sentence analysis, but can be complicated to execute.

df\_ecb %>%  
 make\_ngrams(2) %>%  
 count\_ngrams() %>%  
 visualize\_bigrams(min.n = 100)



## Counting and correlation among sections

We calculated the correlation of words within years to try and identify whether there is a change in how words are used across the years. But there seems to be no useful results.

Another consideration is that using the count of words within each set of speech is not sufficient to accurately reflect correlation. We might want to consider calculating the correlation of words within paragraphs.

ecb\_year\_words <- ecb\_unigrams %>%  
 mutate(section = year(date))  
  
library(widyr)  
  
# Calculating annual correlation  
word\_cors <- bind\_rows(  
 lapply(unique(ecb\_year\_words$section),   
 function(x) {  
 ecb\_year\_words %>%  
 filter(section == x) %>%  
 group\_by(word) %>%  
 filter(n >= 10) %>%  
 pairwise\_cor(word, date, n, sort = TRUE, use = "complete.obs") %>%  
 mutate(section = x)  
}))  
  
word\_cors %>%  
 filter(item1 %in% c("growth", "inflation")) %>%  
 group\_by(item1, section) %>%  
 top\_n(6, desc(abs(correlation))) %>%  
 ungroup() %>%  
 arrange(desc(correlation)) %>%  
 mutate(item2 = reorder(item2, correlation)) %>%  
 filter(section > 2010) %>%  
 ggplot(aes(item2, correlation)) +  
 geom\_bar(stat = "identity") +  
 facet\_wrap(~ section + item1, ncol = 2, scales = "free") +  
 coord\_flip()

