Machine Learning with Applications 8 (2022) 100286

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

# Text mining methodologies with R: An application to central bank texts ☆

Jonathan Benchimol [a,*], Sophia Kazinnik [b], Yossi Saadon [a]

[a] Research Department, Bank of Israel, Jerusalem, Israel
[b] Quantitative Supervision and Research, Federal Reserve Bank of Richmond, Charlotte, NC, USA

ARTICLE INFO

*JEL classification:*
B40
C82
C87
D83
E58

*Keywords:*
Text mining
R programming
Sentiment analysis
Topic modeling
Natural language processing
Central bank communication
Bank of Israel

ABSTRACT

We review several existing text analysis methodologies and explain their formal application processes using the open-source software R and relevant packages. Several text mining applications to analyze central bank texts are presented.

## 1. Introduction

The information age is characterized by the rapid growth of data, mostly unstructured data. Unstructured data is often text-heavy, including news articles, social media posts, Twitter feeds, transcribed data from videos, as well as formal documents.[1] The availability of this data presents new opportunities, as well as new challenges, both to researchers and research institutions. In this paper, we review several existing methodologies for analyzing texts and introduce a formal process of applying text mining techniques using the open-source software R. In addition, we discuss potential empirical applications.

This paper offers a primer on how to systematically extract quantitative information from unstructured or semi-structured text data. Quantitative representation of text has been widely used in disciplines such as computational linguistics, sociology, communication, political science, and information security. However, there is a growing body of literature in economics that uses this approach to analyze macroeconomic issues, particularly central bank communication and financial stability.[2]

The use of this type of text analysis is growing in popularity and has become more widespread with the development of technical tools and packages facilitating information retrieval and analysis.[3]

☆ This paper does not necessarily reflect the views of the Bank of Israel, the Federal Reserve Bank of Richmond or the Federal Reserve System. The present paper serves as the technical appendix of our research paper (Benchimol et al., 2020). We thank the Editor, anonymous referees, Itamar Caspi, Shir Kamenetsky Yadan, Ariel Mansura, Ben Schreiber, and Bar Weinstein for their productive comments. The replication files are available at https://doi.org/10.24433/CO.1132001.v1.

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
*E-mail addresses:* jonathan.benchimol@boi.org.il (J. Benchimol), sophia.kazinnik@rich.frb.org (S. Kazinnik), yosis@boi.org.il (Y. Saadon).

[1] Usually in Adobe PDF or Microsoft Word formats.

[2] See, for instance, Carley (1993), Ehrmann and Fratzscher (2007), Lucca and Trebbi (2009), Bholat et al. (2015), Hansen and McMahon (2016), Bholat et al. (2019), Bruno (2017), Calomiris and Mamaysky (2020), Hansen et al. (2019), Benchimol et al. (2021), Correa et al. (2021), and Ter Ellen et al. (2022).

[3] See, for instance, Lexalytics, IBM Watson AlchemyAPI, Provalis Research Text Analytics Software, SAS Text Miner, Sysomos, Expert System, RapidMiner Text Mining Extension, Clarabridge, Luminoso, Bitext, Etuma, Synapsify, Medallia, Abzooba, General Sentiment, Semantria, Kanjoya, Twinword, VisualText, SIFT, Buzzlogix, Averbis, AYLIEN, Brainspace, OdinText, Loop Cognitive Computing Appliance, ai-one, LingPipe, Megaputer, Taste Analytics, LinguaSys, muText, TextualETL, Ascribe, STATISTICA Text Miner, MeaningCloud, Oracle Endeca Information Discovery, Basis Technology, Language Computer, NetOwl, DiscoverText, Angoos KnowledgeREADER, Forest Rim's Textual ETL, Pingar, IBM SPSS Text Analytics, OpenText, Smartlogic, Narrative Science Quill, Google Cloud Natural Language API, TheySay, indico, Microsoft Azure Text Analytics API, Datumbox, Relativity Analytics, Oracle Social Cloud, Thomson Reuters Open Calais, Verint Systems, Intellexer, Rocket Text Analytics, SAP HANA Text Analytics, AUTINDEX, Text2data, Saplo, SYSTRAN, and many others.

https://doi.org/10.1016/j.mlwa.2022.100286
Received 1 March 2021; Received in revised form 9 March 2022; Accepted 9 March 2022
Available online 25 March 2022

An applied approach to text analysis can be described by several sequential steps. Given the unstructured nature of text data, a consistent and repeatable approach is required to assign a set of meaningful quantitative measures to this type of data. This process can be roughly divided into four steps: data selection, data cleaning, information extraction, and analysis of that information. Our tutorial explains each step and shows how it can be executed and implemented using the open-source R software. For our sample data set, we use a set of monthly communications published by the Bank of Israel.

In general, an automatic and precise understanding of financial texts allows for the construction of relevant financial indicators. There are many potential applications in economics and finance, as well as other social science disciplines. Central bank publications (e.g., interest rate announcements, minutes, speeches, official reports, etc.) are of particular interest, considering what a powerful tool central bank communication is (Benchimol et al., 2020). This quick and automatic analysis of the underlying meaning conveyed by these texts should allow for fine-tuning of these publications before making them public. For instance, a spokesperson could use this tool to analyze the orientation of a text, such as an interest rate announcement, before making it public.

The remainder of the paper is organized as follows. The next section covers theoretical background behind text analysis and interpretation of text. Section 3 describes text extraction and Section 4 presents methodologies for cleaning and storing text data for text mining. Section 5 presents several common approaches to text data structures used in Section 6, which details methodologies used for text analysis, and Section 7 concludes.

## 2. Theoretical background

The principal goal of text analysis is to capture and analyze all possible meanings embedded in the text. This can be done both qualitatively and quantitatively. The purpose of this paper is to offer an accessible tutorial to the quantitative approach. In general, quantitative text analysis is a field of research that studies the ability to decode data from natural language with computational tools.

Quantitative text analysis takes roots in a set of simple computational methods, focused on quantifying the presence of certain keywords or concepts with a text. These methods, however, fail to take into account the underlying meaning of text. This is problematic because, as shown by Carley (1993), two identical sets of words can have very different meanings. This realization and subsequent need to capture meaning embedded in text gave rise to the development of new methods, such as language network models, and, specifically, semantic networks (Danowski, 1993; Diesner, 2013). Today, the common approach in quantitative text mining is to find relationships between concepts, generating what is known as a semantic network.

Semantic network analysis is characterized by its ability to illustrate the relationships between words within a text, providing insights into its structure and meaning. Semantic networks rely on co-occurrence metrics to represent proximity concepts (Diesner, 2013; Diesner & Carley, 2011a, 2011b). For instance, nodes in a network represent concepts or themes that frequently co-occur near each other in a specific text. As a result, semantic network analysis allows meaning to be revealed by considering the relationships among concepts.

In this paper, we cover both of the approaches mentioned above. We first discuss term-counting methods, such as term frequency and relative frequency calculations. We follow with networks-based methods, such as cluster analysis, topic modeling, and latent semantic analysis. Overall, the field of natural language processing (NLP) has progressed rapidly in recent years, but these methods still remain to be essential and relevant building blocks of quantitative language analysis.

The next three sections present a comprehensive set of steps for text analysis, starting with common methodologies for cleaning and storing text, as well as discussing several common approaches to text data structures.

## 3. Text extraction

For this exercise, we use a set of interest rate announcements published by the Bank of Israel from 1997 to 2017. Overall, we have 220 documents of this type. We use this set of documents as input using package `tm` (Feinerer et al., 2008) within the open-source software R.[4] This package can be thought as a framework for text mining applications within R, including text preprocessing. There is a core function called `corpus` embedded in the `tm` package. This function takes a predefined directory, which contains the input (a set of documents) as an argument, and returns the output, which is the set of documents organized in a particular way. Here, we use the term *corpus* to reference a relevant set of documents.

We define our corpus in R in the following way. First, we apply a function called `file.path` that defines a directory where all of our text documents are stored.[5] In our example, it is the folder that stores all 220 text documents, each corresponding to a separate interest rate decision meeting. After defining the working directory, we apply the function `corpus` from the package `tm` to all of the files in the working directory. This way, the function captures and interprets each file as a document and formats the set of text documents into a corpus object class as defined internally by the `tm` package.

```
file.path <- file.path("../data/folder")
corpus <- corpus(DirSource(file.path))
```

Now, we have our documents organized as a corpus object. The content of each document can be accessed and read using the `writeLines` function. For example:

```
writeLines(as.character(corpus[[1]]))
```

Using this command line, we can access and view the content of document number one within the corpus. This document corresponds to the interest rate discussion published in December 2007. Below are the first two sentences of this document:

*Bank of Israel Report to the public of the Bank of Israel's discussions before setting the interest rate for January 2007 The broad-forum discussions took place on December 24, 2006, and the narrow forum discussions on December 25, 2006, January 2007 General Before the Governor makes the monthly interest rate decision, discussions are held at two levels. The first discussion takes place in a broad forum, in which the relevant background economic conditions are presented, including real and monetary developments in Israel's economy and developments in the global economy.*

There are other approaches to storing a set of texts in R, for example by using the functions `data.frame` or `tibble`. However, we will concentrate on `tm`'s corpus approach as it is largely more intuitive, and has a greater number of corresponding functions explicitly written for text analysis.

## 4. Cleaning and storing text

Once the relevant corpus is defined, and read into the program, we can transform it into an appropriate format for further analysis. As mentioned previously, each document can be thought of as a set of tokens. Tokens are sets of words, numbers, punctuation marks, and any other symbols present in the given document.

[4] Unnecessary elements (characters, images, advertisements, etc.) are removed from each document to constitute our clean set of documents.

[5] The folder should contain text documents only. If there are other files in that location (i.e., R files) than the `corpus` function will include the text in the other files.

Not all of the tokens carry meaningful information. Therefore, the next necessary step is text cleaning, one of the crucial steps in text analysis. Text cleaning (or text preprocessing) makes an unstructured set of texts uniform across and within and eliminates idiosyncratic characters or meaningless terms.[6] Text cleaning can be loosely divided into a set of steps as shown below.

The text excerpt presented in Section 3 contains some useful information about the content of the discussion, but also a lot of unnecessary details, such as punctuation marks, dates, ubiquitous words. Therefore, the first logical step is to remove punctuation and idiosyncratic characters from the set of texts. This includes any strings of characters present in the text, such as punctuation marks, percentage or currency signs, or any other characters that are not words.

One way to eliminate punctuation marks is with the `removePunctuation` function. This function removes a set of predefined punctuation characters.

```
corpus <- tm_map(corpus, removePunctuation)
```

The text below shows our original excerpt, with the aforementioned punctuation characters removed:

*The broad forum discussions took place on December 24 2006 and the narrow forum discussions on December 25 2006 January 2007 General Before the Governor makes the monthly interest rate decision discussions are held at two levels The first discussion takes place in a broad forum in which the relevant background economic conditions are presented including real and monetary developments in Israel's economy and developments in the global economy*

Additionally, any numbers present in the texts of our corpus can be removed using the `removeNumbers` function, as in the below code:

```
corpus <- tm_map(corpus, removeNumbers)
```

Now, the text below shows our original excerpts, but without any punctuation marks or digits:

*The broad forum discussions took place on December and the narrow forum discussions on December January General Before the Governor makes the monthly interest rate decision discussions are held at two levels The first discussion takes place in a broad forum in which the relevant background economic conditions are presented including real and monetary developments in Israels economy and developments in the global economy*

The current text excerpt is slightly more concise than the original, but there is still much unnecessary information. Therefore, the next step would be to remove the so-called stop words from the text.

What are stop words? Words such as "the", "a", "and", "they", and many others can be defined as stop words. Stop words usually refer to the most common words in a language, and because they are so common, they do not convey specific information. Since these terms do not carry any meaning as standalone terms, they are not valuable for our analysis. In addition to a pre-existing list of stop words, ad hoc stop words can be added to the list.

We apply a function from the package `tm` onto our existing corpus as defined above in order to remove the stop words. There is a coercing function called `removeWords` that erases a given set of stop words from the corpus. There are different lists of stop words available, and we use a standard list of English stop words.

However, before removing the stop words, we need to turn all of our existing words within the text into lowercase. Why? Because converting to lowercase, or case folding, allows for case-insensitive comparison. This is the only way for the function `removeWords` to identify the words subject for removal.

Therefore, using the package `tm`, and a coercing function `tolower`, we convert our corpus to lowercase:

```
corpus <- tm_map(corpus, tolower)
```

Below is the example text excerpt following the command mentioned above:

*the broad forum discussions took place on december and the narrow forum discussions on december january general before the governor makes the monthly interest rate decision discussions are held at two levels the first discussion takes place in a broad forum in which the relevant background economic conditions are presented including real and monetary developments in israels economy and developments in the global economy*

We can now remove the stop words from the text:

```
corpus <- tm_map(corpus, removeWords,
stopwords("english"))
```

Here, `tm_map` is a wrapper function that takes the corpus and applies character processing function `removeWords` onto all of the contents of the corpus (all 220 documents). It returns the modified documents in the same format, a corpus, but with the changes already applied. The following output shows our original text excerpt with the stop words removed:

*broad forum discussions took place december narrow forum discussions december january general governor makes monthly interest rate decision discussions held two levels first discussion takes place broad forum relevant background economic conditions presented including real monetary developments israels economy developments global economy*

The next and final step is to stem the remaining words. Stemming is a process of turning several words that mean the same thing into one. For example, after stemming, words such as "banking", "banks", and "banked" become "bank". As stemming reduces inflected or derived words to their word stem or root. This can be thought of as word normalization. Stemming algorithm allows us not to count different variations as the same term as separate instances.

Below, we use a coercing function called `stemDocument`, that stems words in a text document using Porter's stemming algorithm (Porter, 1980). This algorithm removes common morphological and inflectional endings from words in English, as described in the previous paragraph.

```
corpus <- tm_map(corpus, stemDocument)
```

Once we have applied several of these character processing functions to our corpus, we would like to examine it in order to view the results. Overall, as a result of the above procedures, we end up with the following:

*broad forum discuss took place decemb narrow forum discuss decemb januari general governor make month interest rate decis discuss held two level first discuss take place broad forum relev background econom condit present includ real monetari develop israel economi develop global economi*

This last text excerpt shows what we end up with once the data cleaning manipulations are done. While the excerpt we end up with resembles its original only remotely, we can still figure out reasonably well the subject of the discussion.

[6] Specific characters that are not used to understand the meaning of a text.

**Table 1**
Document Term Matrix - Excerpt.

| Document $i$ | Term $j$ | | | |
|---|---|---|---|---|
| | Accord | Activ | Averag | ... |
| May 2008 | 3 | 9 | 4 | ... |
| June 2008 | 6 | 4 | 16 | ... |
| July 2008 | 5 | 3 | 7 | ... |
| August 2008 | 4 | 9 | 12 | ... |
| September 2008 | 5 | 8 | 22 | ... |
| October 2008 | 3 | 20 | 16 | ... |
| November 2008 | 6 | 5 | 11 | ... |
| ... | ... | ... | ... | ... |

*Note*: This table presents an excerpt of a document term matrix (`dtm`).

## 5. Data structures

Once the text cleaning step is done, R allows for several convenient ways of storing our results. We discuss the two most popular formats, a document term matrix and a `tidy` object. While there may be more ways to store text, these two formats are the most convenient when working with text data in R. We explain each of these formats next.

### 5.1. Document term matrix

Document term matrix is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. Such matrices are widely used in the field of NLP. In a document term matrix, each row corresponds to a specific document in the collection and each column corresponds to the count of specific terms within that document. In essence, a document term matrix is a vector-space representation of a set of documents. An example of a document term matrix is shown in Table 1.

This type of matrix represents the frequency for each unique term in each document in corpus. In R, our corpus can be mapped into a `dtm` object class by using the function `DocumentTermMatrix` from the `tm` package.

```
dtm <- DocumentTermMatrix(corpus)
```

The goal of mapping the corpus onto a `dtm` is twofold; the first is to present the topic of each document by the frequency of semantically significant and unique terms, and second, to position the corpus for future data analysis. The value in each cell of this matrix is typically the word frequency of each term in each document. This frequency can be weighted in different ways. The default weighting scheme function is called term frequency (`tf`).

Another common approach to weighting is called term frequency–inverse document frequency (`tf-idf`). While the `tf` weighting scheme is defined as the number of times a word appears in the document, `tf-idf` is defined as the number of times a word appears in the document but is offset by the frequency of the words in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

Why is the frequency of each term in each document important? A simple counting approach such as term frequency may be inappropriate because it can overstate the importance of a small number of very frequent words. Term frequency is the most normalized one, measuring how frequently a term occurs in a document with respect to the document length, such as:

$$\texttt{tf}(t) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}} \tag{1}$$

A more appropriate way to calculate word frequencies is to employ the `tf-idf` weighting scheme. It is a way to weight the importance of terms in a document based on how frequently they appear across multiple documents. If a term frequently appears in a document, it is important, and it receives a high score. However, if a word appears in many documents, it is not a unique identifier, and it will receive a low score. Eq. (1) shows how words that frequently appear in a single document will be scaled up, and Eq. (2) shows how common words which appear in many documents will be scaled down.

$$\texttt{idf}(t) = \ln\left(\frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}}\right) \tag{2}$$

Conjugating these two properties yield the `tf-idf` weighting scheme:

$$\texttt{tf} - \texttt{idf}(t) = \texttt{tf}(t) \times \texttt{idf}(t) \tag{3}$$

In order to employ this weighting scheme in Eq. (3), we can assign this option within the already familiar function `dtm`:

```
dtm <- DocumentTermMatrix(corpus,
control = list(weighting = weightTfIdf))
```

The above-mentioned steps provide us with a suitable numeric matrix under the name of `dtm`. In this matrix, each cell contains an integer, corresponding to a (perhaps weighted) number of times a specific term appeared in each document within our corpus. However, most cells in such `dtm` will be empty, i.e., zeros, because most terms do not appear in most of the documents. This property is referred to as matrix sparsity.

Most term-document matrices tend to be high-dimensional and sparse. This is because any given document will contain only a subset of unique terms that appear throughout the corpus. This will result in any corresponding document-row having zeros for terms that were not used in that specific document. Therefore, we need an approach to reduce dimensionality.

Function `RemoveSparseTerms` takes our existing `dtm`, and a certain numerical value called `sparse`, and returns an adjusted `dtm`, where the number of elements is reduced, depending on the value set for `sparse`. This numerical value is defined as the maximum allowed sparsity, as is set in the range of zero to one.

In reference to `RemoveSparseTerms`, sparsity refers to the threshold of relative document frequency of a term, above which the term will be removed. Sparsity becomes smaller as this threshold approaches one. In a way, this process can be thought of as removing outliers. For example, the code below will yield a `dtm` where all terms appearing in at least 90% of the documents will be kept.

```
dtm <- RemoveSparseTerms(dtm, 0.1)
```

Now, we have a `dtm` that is ready for the initial text analysis. An example of output following this weighting scheme and subsequent sparsity reduction of a certain degree might yield Table 2.

### 5.2. Tidytext table

This section provides an overview of the `tidytext` R package, detailed in Wickham (2014). This framework was developed specifically for the R software, and for the sole purpose of text mining. `tidytext` allows a set of documents to be presented as a one-term-per-document-per-row data frame. This is done with the help of the `unnest_tokens` function within the `tidytext`.

This one-observation-per-row structure contrasts the way text is often stored in alternative frameworks.[7] For `tidytext` , the observation (or token) that is stored in each row is most often a single term, but can

[7] This is different from other formats where each word corresponds with the document from which it comes. In addition, it is possible to convert the `tidytext` format into the `dtm` format.

**Table 2**
Document Term Matrix — Excerpt with `tf-idf`.

| | abroad | acceler | accompani | account | achiev | adjust | ... |
|---|---|---|---|---|---|---|---|
| **01-2007** | 0.0002 | 0.000416 | 0.000844 | 0.000507 | 0.000271 | 0.000289 | ... |
| **01-2008** | 0.00042 | 0.000875 | 0.000887 | 0.000152 | 9.49E-05 | 0.000304 | ... |
| **01-2009** | 0.000497 | 0 | 0 | 9.01E-05 | 0.000112 | 0.000957 | ... |
| **01-2010** | 0.000396 | 0 | 0 | 7.18E-05 | 8.95E-05 | 0.000954 | ... |
| **01-2011** | 0.000655 | 0 | 0.000691 | 0.000119 | 7.39E-05 | 0.000552 | ... |
| **01-2012** | 0.000133 | 0 | 0.001124 | 9.65E-05 | 6.01E-05 | 0 | ... |
| **01-2013** | 0.00019 | 0.000395 | 0 | 0.000138 | 8.56E-05 | 0.000274 | ... |
| **01-2014** | 0 | 0.000414 | 0 | 0.000144 | 8.98E-05 | 0 | ... |
| **01-2015** | 0 | 0.00079 | 0 | 6.88E-05 | 8.57E-05 | 0.000183 | ... |
| **01-2016** | 0 | 0.000414 | 0 | 0 | 0.00018 | 0.000192 | ... |
| **01-2017** | 0 | 0.000372 | 0.000755 | 6.48E-05 | 0.000323 | 0.000689 | ... |
| **01-2018** | 0.000581 | 0 | 0.002455 | 0.000211 | 0 | 0 | ... |

*Notes*: This table presents an excerpt of a `dtm` with `tf-idf` weighting methodology. The highest values for the selected sample are highlighted in gray.

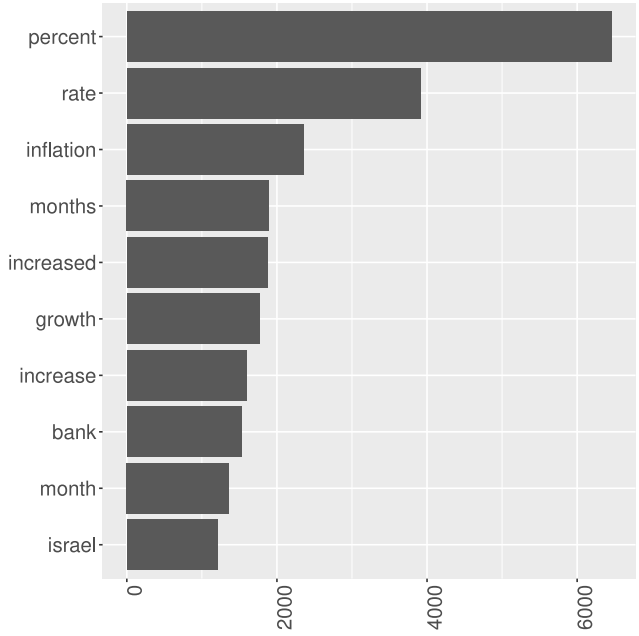


**Fig. 1.** Histogram - `tidytext`.
*Note*: Histogram containing most popular terms within the `tidytext` table.

also be an n-gram, sentence, or paragraph. In general, a token refers to a unit by which we analyze the text.

Tokenization by word is one of the most intuitive units of analysis, but there are cases where we may want to break the data into other units. For example, an n-gram is an adjacent sequence of n terms from a given sample of text. One might choose this unit when it is important to capture certain expressions, or sets of words that go together.

For example, Fig. 1 presents the most frequent words in the corpus as produced by the `tidytext` package. The below code selects words that appear in the corpus at least 1200 times or more and plots their frequencies. Here, `n` is the word count, i.e., how many times each word appears in the corpus:

```
tidy.table <- tidy.table %>%
count(word, sort = TRUE) %>%
filter(n > 1200) %>%
mutate(word = reorder(word, n)) %>%
ggplot(aes(word, n)) + geom_col() + xlab(NULL)
+ coord_flip()
```

Besides being more intuitive, the `tidytext` package has the capability for better graphics.

### 5.3. Data exploration

Given a `dtm` with reduced dimensions, as described above, we can apply exploratory analysis techniques to find out what kind of information the corpus as a whole, or each document within the corpus, contains. As with the text cleaning procedures, there are several logical steps, and the first would be to find out what are the most frequent terms within the `dtm`.

The following piece of code sums up the columns within the `dtm`, and then sorts them in descending order within a data frame. We can then view terms with the six highest and the lowest frequencies by using functions `head` and `tail`, respectively:

```
term.frequencies < - colSums(as.matrix(dtm))
order.frequencies < - order(term.frequencies)
head(order.frequencies, 6)
tail(order.frequencies, 6)
```

Table 3 is an example of an output following these commands, showing the top six most frequent words and their corresponding frequencies.

Another, and perhaps easier, way to identify the frequency terms within the `dtm` is to use the function `findFreqTerms`, which is a part of the package `tm`. This function returns a list of terms, which meet two ad-hoc criteria of upper and lower bounds of frequency limits:

```
findFreqTerms(dtm, lowfreq = 1800, highfreq = 5000)
```

Here is an example of an output following these commands. Fig. 2 shows a histogram of the terms that appear at least 1800 within our corpus.

Another way to look at the most popular terms is to use the `dtm` with the `tf-idf` frequency weighted terms. Fig. 3 shows a histogram of some of the most common terms within the corpus, as weighted by the `tf-idf` approach.

Once we know some of the most frequent terms in our corpus, we can explore the corpus further by looking at different associations between some of them. This can be done with the function `findAssocs`, part of the package `tm`. This function takes our `dtm`, and a user-selected specific term such as “bond”, “increas”, “global”, etc., and an inclusive lower correlation limit as an input, and returns a vector of matching terms and their correlations (satisfying the lower correlation limit `corlimit` ):

```
findAssocs(dtm, "bond", corlimit = 0.5)
findAssocs(dtm, "econom", corlimit = 0.35)
findAssocs(dtm, "fed", corlimit = 0.35)
```

Table 4 is an example of an output following these for the term “bond”.

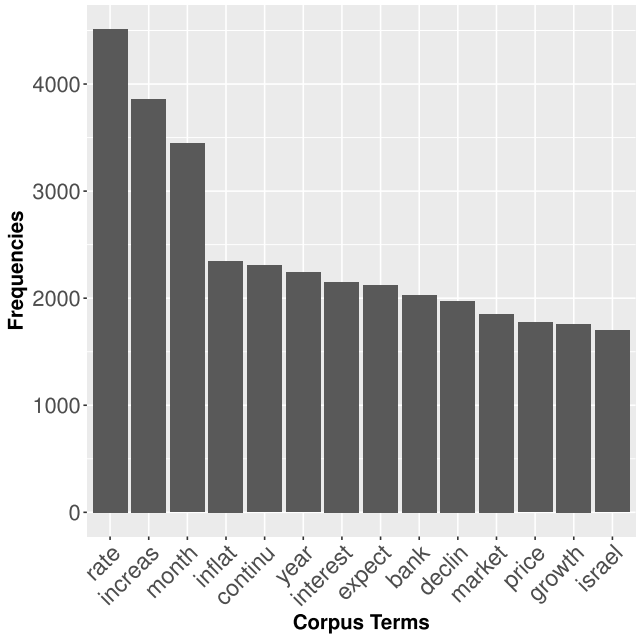


**Fig. 2.** Histogram - Corpus.
*Note*: Histogram containing most popular terms within the corpus.

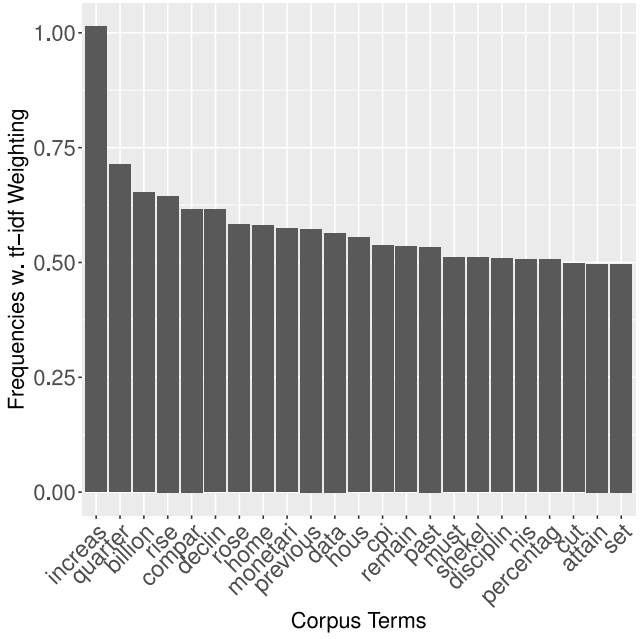


**Fig. 3.** Histogram - `tf-idf`.
*Note*: Histogram containing most popular terms within the corpus, with `tf-idf` weighting.

While this might not be the best way to explore the content of each text or the corpus in general, it might provide some interesting insights for future analysis. The math behind the `findAssocs` function is based on the standard function `cor` in R's `stats` package. Given two numeric vectors, `corlimit` computes their correlation.

Another way to explore the contents of our corpus is to create a so-called word cloud. A word cloud is an image composed of words used in a particular text or corpus, in which the size of each word indicates its frequency. This can be done with the use of the `wordcloud` package.[8] Below, we plot word clouds using two different approaches to calculating the frequency of each term in the corpus. The first approach uses a simple frequency calculation. We use the function `set.seed` here to ensure all results and figures are reproducible. We can also vary the `min.freq` parameter to select the minimum frequency threshold of each term.

**Table 3**
Top 6 Frequent Words in `dtm`.

| Term $j$ | |
|---|---|
| Percen | 9927 |
| Rate | 9127 |
| Increas | 5721 |
| Month | 5132 |
| Interest | 4861 |
| Bank | 4039 |

*Note*: top six most frequent words with their corresponding frequencies.

**Table 4**
Correlation - `bond`.

| Term $j$ | |
|---|---|
| Yield | 0.76 |
| Month | 0.62 |
| Market | 0.61 |
| Credit | 0.60 |
| Aviv | 0.58 |
| Rate | 0.58 |
| Bank | 0.57 |
| Indic | 0.57 |
| Announc | 0.56 |
| Sovereign | 0.55 |
| Forecast | 0.55 |
| Held | 0.54 |
| Measur | 0.54 |
| Treasuri | 0.52 |
| Germani | 0.52 |
| Index | 0.51 |

*Note*: Terms most correlated with the term "bond".

```
set.seed(123)
wordcloud(names(freq), freq, min.freq = 400,
colors=brewer.pal(8, "Dark2"))
```

The function `wordcloud` provides a nice and intuitive visualization of the content of the corpus, or if needed, of each document separately. Figs. 4 and 5 provide several examples of an output following these commands. For instance, Figs. 4 and 5 show word clouds containing word terms that appear at least 700, 1000, and 2000, respectively.

Alternatively, to explore the frequency of terms within the corpus, one can use the `tf-idf` weighting scheme, and reproduce the figures. It is clear that with the new weighting scheme, other terms are emphasized more. As an example, Figs. 6 and 7 show word clouds with word frequencies above 0.06, 0.08 and 0.1.

Another way to explore relationships between terms is to employ network analysis, as it allows for a more comprehensive approach to exploring relationships within the corpus terms. In general, network analysis allows texts to be modeled as networks, with connections between nodes representing relations between them. A common way to model text in this framework is to apply a clustering algorithm to visualize it with a type of dendrogram or adjacency diagram. We provide examples for both types next.

Clustering algorithms provide a way of identifying clusters of nodes that are densely connected to each other on the network. The clustering method can be thought of as an automatic text categorization. The basic

[8] https://cran.r-project.org/web/packages/wordcloud/wordcloud.pdf

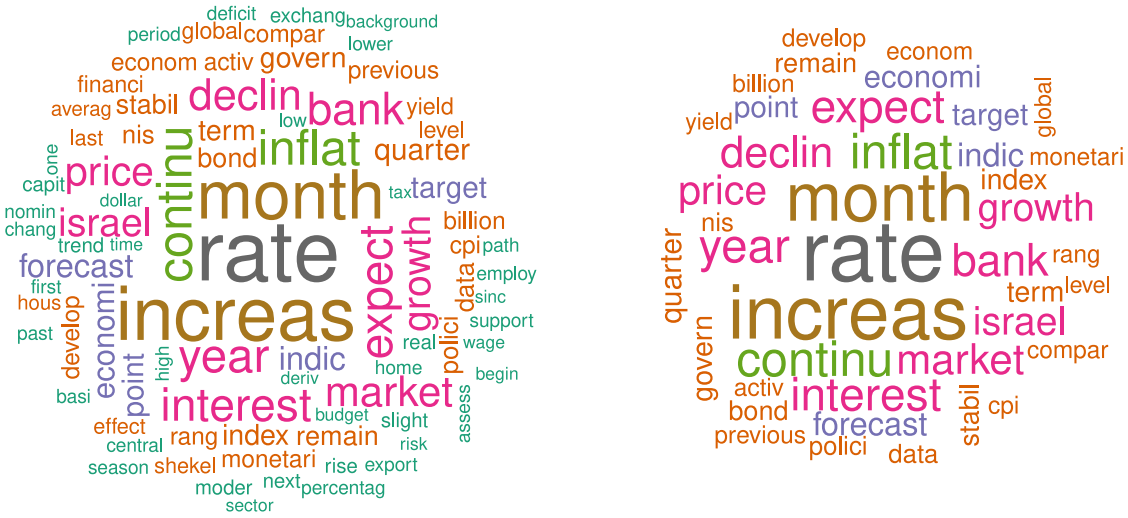


**Fig. 4.** Word Cloud - 400 and 700 Terms.
*Note*: Word cloud containing terms that appear at least 400 (left panel) and 700 (right panel) times in the corpus, with word frequencies above 0.06.

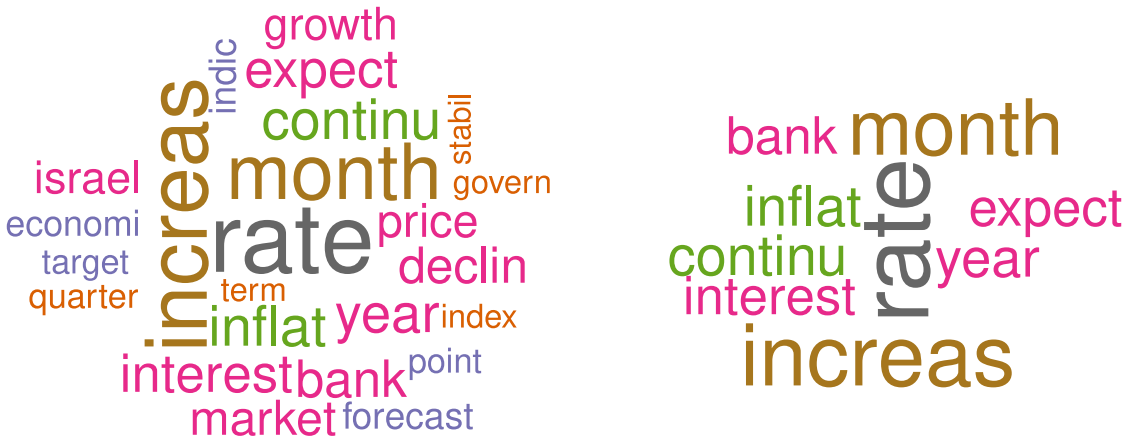


**Fig. 5.** Word Cloud - 1000 and 2000 Terms.
*Note*: Word cloud containing terms that appear at least 1000 and 2000 times in the corpus (left and right panel, respectively), with word frequencies above 0.06.

idea behind document or text clustering is to categorize documents into groups based on likeness criteria. For example, one could calculate the Euclidean, or geometric, distance between the terms to get a measure of likeness. Terms are then grouped on some distance related criteria.

One of the most intuitive ways to visualize relationships between terms is with correlation maps. Correlation maps show how some of the most frequent terms relate to each other in the corpus, based on a certain *ad-hoc* correlation criteria. While this is not a directed graph, like ones used in semantic networks, it still reflects some information about the relationships between terms.

Below is the code example that can create a correlation map for a given `dtm`. To plot this object, one will need to use the `Rgraphviz` package.[9]

```
correlation.limit < - 0.6
freqency.terms.dtm < - findFreqTerms(dtm.tf.idf)
plot(dtm.tf.idf, term = freqency.terms.dtm,
corThreshold=correlation.limit)
```

[9] Available at: http://bioconductor.org/biocLite.R or through R console such as:

```
install.packages("BiocManager")
BiocManager::install("Rgraphviz")
```

We plot Figs. 8 and 9 following the above code.

Another way to visualize a network representing relationships between terms within the corpus is to create a dendrogram. This technique arranges the network into a hierarchy of groups according to a specified method. Hierarchical clustering is an approach that allows to identifying groups in the data set without pre-specifying the number of clusters. Hierarchical clustering output can be visualized in a tree-based representation of the observations, called a dendrogram. We can plot a dendogram for our `dtm` using the following commands:

```
dendogram < - dist(t(dtm.sparse.01),
method = "euclidian")
dendogram.fit < - hclust(d = dendogram,
method = "ward.D")
plot(dendogram.fit, hang = -1)
```

Here, the clustering method is based on the Ward's method, which is based on the original function approach. It relies on choosing a minimum variance criterion that would minimize the total within-cluster variance (Ward, 1963).

The following dendograms presented in Fig. 10 are built based on `tf` and `tf-idf` weighting schemes, respectively.

Another quick and useful visualization of each document's content is allowed by heat maps. Heat maps can be used to compare the content


**Fig. 6.** Word Cloud - Word Frequencies Above 0.06.
*Note*: Word cloud containing word terms with word frequencies above 0.06.

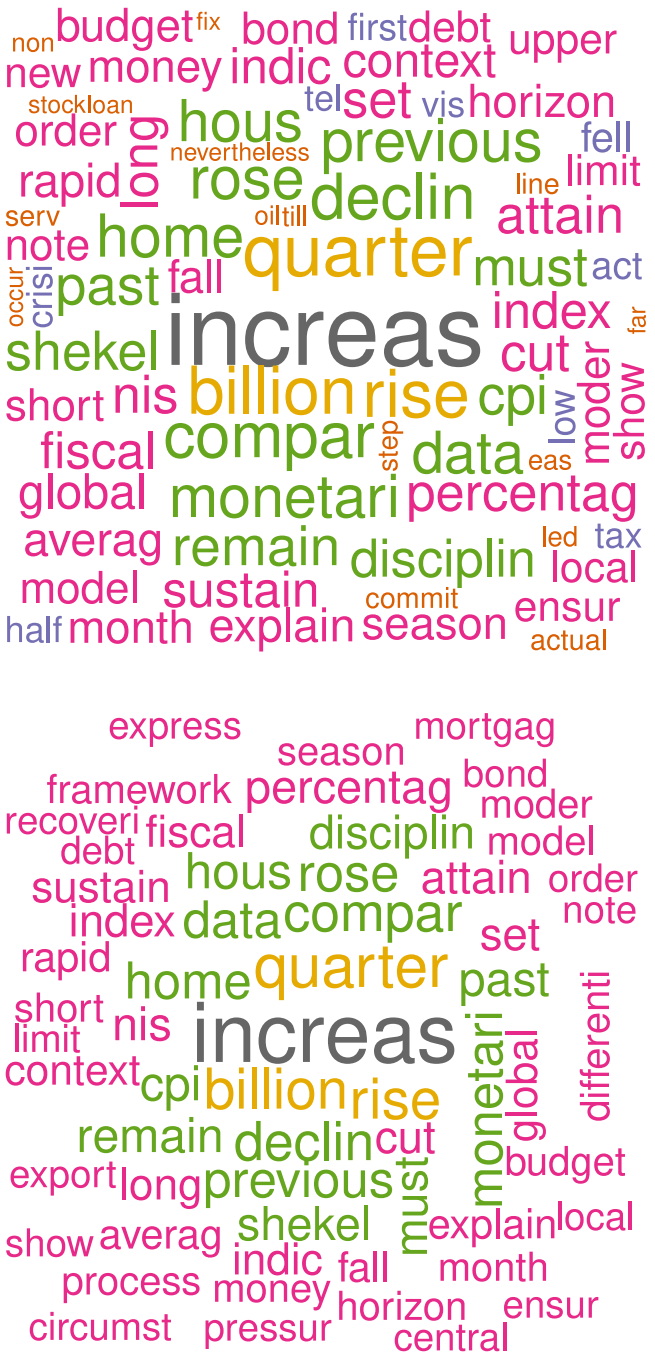


**Fig. 7.** Word Cloud - Word Frequencies Above 0.08.
*Note*: Word cloud containing word terms with word frequencies above 0.08 (top panel) and 0.1 (bottom panel).

of each document, side by side, with other documents in the corpus, revealing interesting patterns and time trends.

Fig. 11 presents word frequencies for the word list on the bottom of the heatmap. It demonstrates a simple distribution of word frequencies throughout time. For example, the term `accommod` was used heavily during the discussions that took place in mid and late 2001; however, it was not mentioned at all in early 2002.

Fig. 12 presents a heatmap of word frequencies for the period spanning the mid-1999 to early 2000. For example, the term "inflat", representing discussion around inflation, shows that this topic was discussed heavily in early 2000, in particular in January of 2000. These kinds of figures provide a quick and visual representation of any given interest rate discussion.

This section sums up some of the most popular techniques for exploratory text analysis, demonstrating how a set of texts can be summarized and visualized in an easy and intuitive way. In the next section, we talk about deriving text-based quantitative metrics, allowing us to measure and summarize various aspects of text.

## 6. Text analytics

In this section we implement methods that allow us to conduct analysis both within and between texts. Techniques such as sentiment analysis and term frequency weighting can be used for the first purpose. The second group is used for comparison between texts and refers to techniques related to Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). We describe these techniques in more detail in this section. The majority of text analytic algorithms in R are written with the `dtm` format in mind. For this reason, we will use `dtm` format in order to discuss the application of these algorithms.

### 6.1. Word counting

Dictionary-based text analysis is a popular technique due to its simplicity. This technique is based on selecting a predefined list of words that are relevant for the analysis of that particular text. For example, the most commonly used source for word classifications in the literature is the Harvard Psycho-sociological Dictionary, specifically, the Harvard-IV-4 TagNeg (H4N) file. In general, dictionaries contain lists of words that correspond to different categories, such as positive or negative sentiment.

However, word categorization for one discipline (for example, psychology) might not translate effectively into another discipline (for example, economics or finance). Therefore, one of the drawbacks of this approach is the importance of adequately choosing an appropriate dictionary or a set of predefined words. Loughran and Mcdonald (2011) demonstrate that some words that may have a negative connotation in one context may be neutral in others. The authors show that dictionaries containing words like tax, cost, or liability that convey negative sentiment in a general context, are more neutral in tone in the context

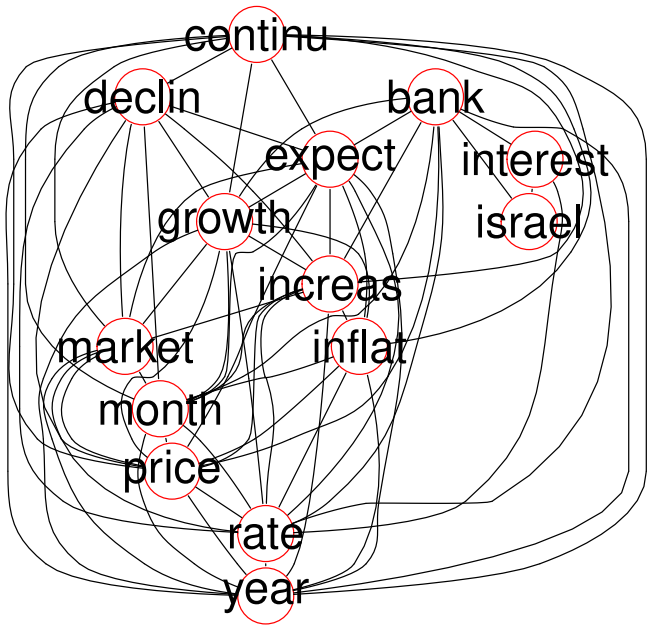


**Fig. 8.** Correlation Map Using `dtm` - Simple Counting Weighting Scheme.
*Note*: The vertical axis represents the heights.

of financial markets. The authors construct an alternative, finance-specific dictionary to reflect tone in a financial text better. They show that, with the use of a finance-specific dictionary, they can predict asset returns better than other, generic, dictionaries.

In this tutorial, we use the Loughran and Mcdonald (2011) master dictionary, which is available on their website. We divide the dictionary into two separate csv files into two sentiment categories. Each file contains one column with several thousand words; one is a list of positive terms, and one is a list of negative terms. We read in both of these files into R as csv files.

```
dictionary.finance.negative <- read.csv(
"negative.csv", stringsAsFactors = FALSE)[,1]
dictionary.finance.positive <- read.csv(
"positive.csv", stringsAsFactors = FALSE)[,1]
```

For example, a word is a positive term if it belongs to a positive, or "hawkish" category: for example, "increas", "rais", "tight", "pressur", "strength", and so on. A word is a negative term if it belongs to a negative or "dovish" category: for example, "decreas", "lower", "loos", "unsatisfi", "worse", and so on. A document is classified as positive if the count of positive words is greater than or equal to the count of negative words. Similarly, a document is classified as negative if the count of negative words is greater than the count of positive words.

The constructed indicator is presented in Fig. 14. We transform both files into two separate data frames, using function `data.frame`. This function is used for storing data tables in a matrix form. We apply the same text cleaning manipulation to the dictionary words as applied to the corpus texts themselves. The code below applies the text data cleaning principles to the two sentiment dictionaries that we have uploaded. The cleaning involves turning all terms within both dictionaries to lowercase, stemming all of the terms, and dropping all duplicate terms:

```
dictionary.negative <-
tolower(dictionary.negative)
dictionary.negative <-
stemDocument(dictionary.negative)
dictionary.negative <-
unique(dictionary.negative)
```

We then use the `match` function that compares the terms in both dictionaries with each term in the corpus. This function returns a value of one if there is a match, and a value of zero if there is no match. This allows us to calculate the number of times each positive and each negative term appeared in the corpus. We proceed to calculate the relative frequency of each dictionary terms. The code below captures the list of terms from the `dtm` by using the function `colnames` and then matches each of the terms in the corpus with the terms in each of the dictionaries, calculating the total amount of matches for each dictionary:

```
corpus.terms <- colnames(dtm)
positive.matches <-
match(corpus.terms, dictionary.positive, nomatch=0)
negative.matches <-
match(corpus.terms, dictionary.negative, nomatch=0)
```

We then assign a value of one to each positive term ($P$) in the document, and a value of minus one to each negative term ($N$) in a document, and measure the overall sentiment score for each document $i$ by the following formula:

$$Score_i = \frac{P_i - N_i}{P_i + N_i} \in [-1; 1] \tag{4}$$

A document is classified as positive if the count of positive words is greater than or equal to the count of negative words. Similarly, a document is negative if the count of negative words is greater than the count of positive words. The code below demonstrates a simple calculation of this indicator:

```
document.score =
sum(positive.matches) - sum(negative.matches)
scores.data.frame =
data.frame(scores = document.score)
```

Fig. 13 presents the main indicators constructed using the dictionary word count.

Using the positive and negative sentiment indicators exposed in Fig. 13, Fig. 14 shows the simple dictionary based sentiment indicator.

Fig. 15 demonstrates a distribution of positive and negative matches throughout the corpus, as produced by the package `tidytext`.

To sum it up, word counting is a relatively straightforward way to summarize the sentiment of any given document. The strength of this approach is that it is intuitive, and easy to implement. In addition, any given dictionary that is being used for document scoring can be customized with *ad-hoc* words, related to the subject matter. This, however, opens the door to a potential weakness of this approach. There is a point where a customized dictionary list might lose its objectivity. Dictionary-based sentiment measurement is the first step in the sentiment extraction process, but there are more sophisticated ways to capture sentiment. We discuss these in the next section.

### 6.2. Relative frequency

In this section we discuss text mining techniques that take into account relative word usage within documents to reveal information about text. Specifically, we discuss a semi-supervised algorithm called `wordscores`, that estimates policy positions by comparing sets of texts using the underlying relative frequency of words. This approach, described by Laver et al. (2003), proposes an alternative way to locate the policy positions of political actors by analyzing the texts they generate. Mainly used in political sciences, it is a statistical technique for estimating the policy position based on word frequencies. The underlying idea is that relative word usage within documents should reveal information of policy positions.

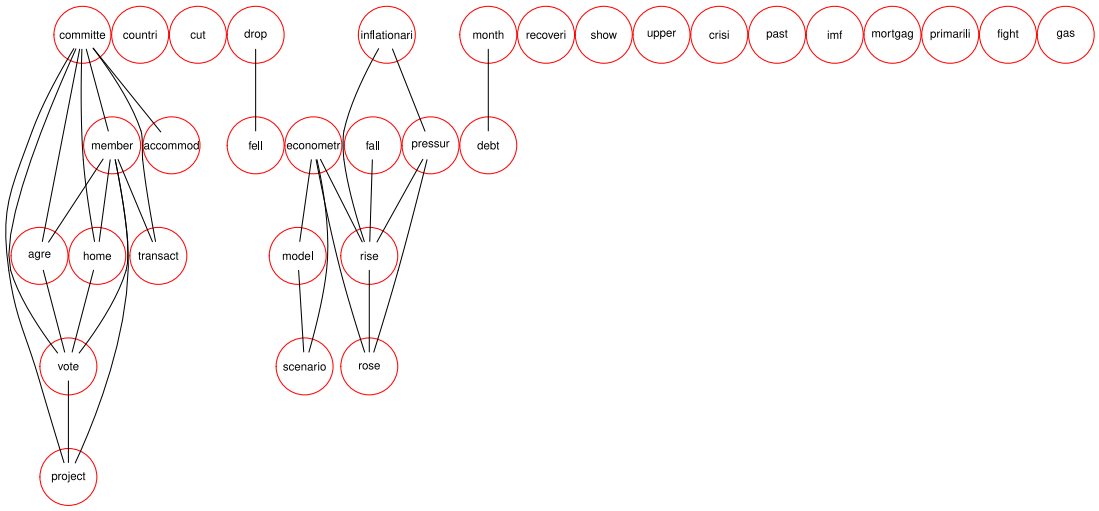


**Fig. 9.** Correlation Map Using `dtm - tf-idf` Frequency.
*Note*: The vertical axis represents the heights.

The algorithm assigns policy positions (or "scores") to documents on the basis of word counts and known document scores (reference texts) via the computation of "word scores". One assumption is that their corpus can be divided into two sets (Laver et al., 2003). The first set of documents has a political position that can be either estimated with confidence from independent sources or assumed uncontroversial. This set of documents is referred to as the "reference" texts. The second set of documents consists of texts with unknown policy positions. These are referred to as the "virgin" texts. The only thing known about the virgin texts is the words in them, which are then compared to the words observed in reference texts with known policy positions.

One example of a reference text describes the interest rate discussion meeting that took place on November 11, 2008. We chose this text as a reference because it is a classic representation of dovish rhetoric. The excerpt below mentions a negative economic outlook, both in Israel and globally, and talks about the impact of this global slowdown on real activity in Israel:

*Recently assessments have firmed that the reduction in global growth will be more severe than originally expected. Thus, the IMF significantly reduced its growth forecasts for 2009: it cut its global growth forecast by 0.8 percentage points to 2.2 percent, and its forecast of the increase in world trade by 2 percentage points, to 2.1 percent. These updates are in line with downward revisions by other official and private-sector entities. The increased severity of the global slowdown is expected to influence real activity in Israel. The process of cuts in interest rates by central banks has intensified since the previous interest rate decision on 27 October 2008.*

Another example of the reference text describes the interest rate discussion meeting that took place on June 24, 2002. This text is a classic representation of hawkish rhetorics. For example, the excerpt below mentions a sharp increase in inflation and inflation expectations:

*The interest-rate hike was made necessary because, due to the rise in actual inflation since the beginning of the year and the depreciation of the NIS, inflation expectations for the next few years as derived from the capital market, private forecasters, and the Bank of Israel's models have also risen beyond the 3 percent rate which constitutes the upper limit of the range defined as price stability. Despite the two increases in the Bank of Israel's interest rate in June, inflation expectations for one year ahead have risen recently and reached 5 percent.*

Specifically, the authors use relative frequencies observed for each of the different words in each of the reference texts to calculate the probability that we are reading a particular reference text, given that we are reading a particular word. This makes it possible to generate a score of the expected policy position of any text, given only the single word in question.

Scoring words in this way replaces and improves upon the predefined dictionary approach. It gives words policy scores, without having to determine or consider their meanings in advance. Instead, policy positions can be estimated by treating words as data associated with a set of reference texts.[10]

In our analysis, out of the sample containing 220 interest rate statements, we pick two reference texts that have a pronounced negative (or "dovish") position and two reference texts that have a pronounced positive (or "hawkish") position regarding the state of the economy during the corresponding month. We assign the score of minus one to the two "dovish" reference texts and the score of one to the two "hawkish" reference texts. We use these known scores to infer the score of the virgin, or out of sample, texts. Terms contained by the out of sample texts are compared with the words observed in reference texts, and then each out of sample text is assigned a score, $Wordscore_i$.

In R, we utilize the package `quanteda`, which contains the function `wordfish`. This function takes a predefined corpus and applies the `wordscores` algorithm as described above. Once the selection process of the reference documents is complete, the code is fairly simple.

```
wordscore.estimation.results <- wordfish(corpus,
dir = c(1,5))
```

This function takes our corpus as an input, as well as the two selected reference documents (here, document number one and document number five), and returns a set of estimation position, as related to each document.

### 6.3. Latent semantic analysis

Latent semantic analysis (LSA) is an unsupervised machine learning algorithm that allows for mapping texts into a vector space (Deerwester

[10] However, one must consider the possibility that there would be a change in rhetoric over time. Perhaps it would make sense to re-examine the approach at certain points in time. This would depend on the time span of the data.

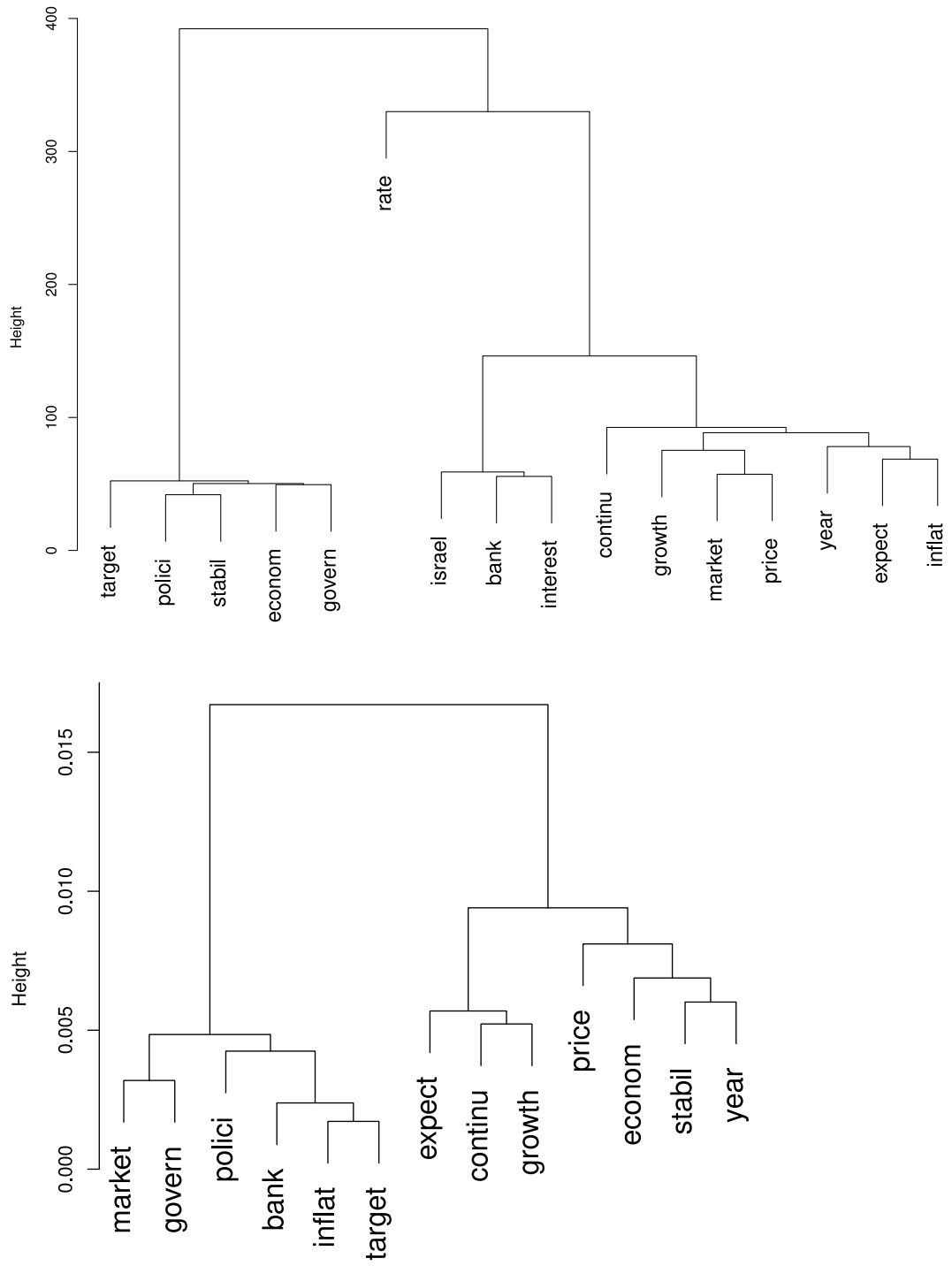


**Fig. 10.** Dendogram - `tf` and `tf-idf` Frequency.
*Notes*: Dendograms for `tf-idf` (top) and `tf-idf` (bottom) frequencies. The vertical axes represent the heights.

et al., 1990). LSA evaluates documents in order to find underlying meaning or concept of these documents. LSA can be used for many things, such as comparing document similarity or clustering documents based on an ad-hoc criterion. LSA can decompose a document term matrix into a reduced vector space that is assumed to reflect semantic structure.

LSA is similar to approaches described in the previous section, but one of its advantages is that it is fully unsupervised, and therefore does not rely on any subjective elements. In R, one can implement LSA with the use of `quanteda` package, thus creating a representation of documents in the reduced 2-dimensional space:

```
lsa_model < - textmodel_lsa(dtm)
lsa_predict < - predict(lsa_model)
```

The output of an estimated LSA model allows for easier comparison between documents, as well as mapping capabilities. Overall, there are various downstream tasks that can be implemented once the latent semantic space is created.

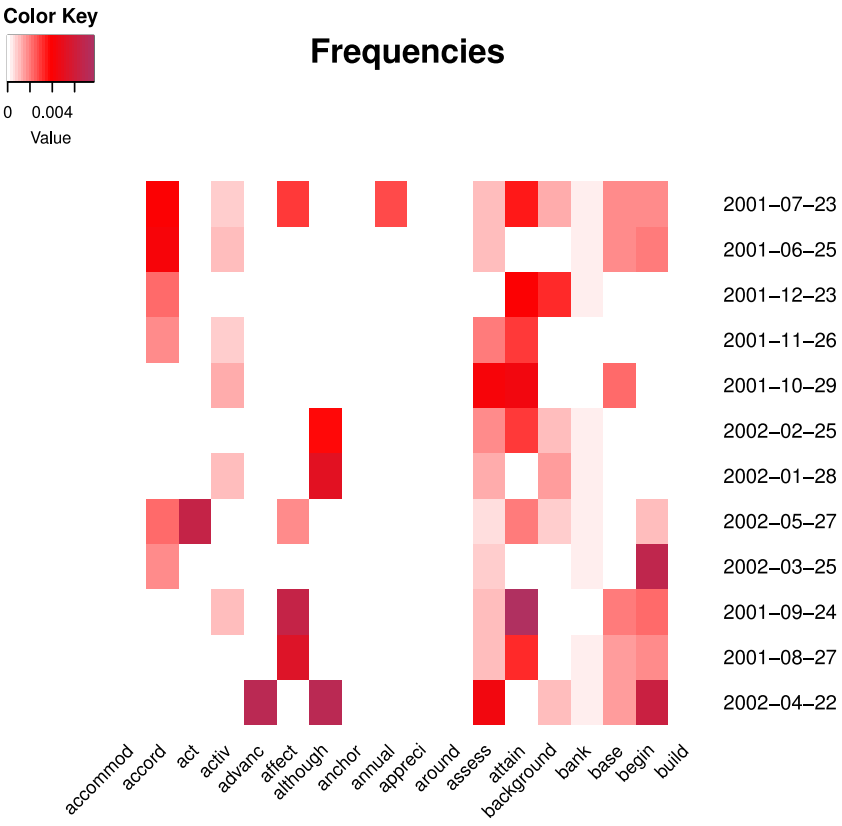


**Fig. 11.** Heatmap `tf-idf` for 2010.
*Note*: Heatmap for documents published in 2010 (`tf-idf` weighted `dtm`). The color key corresponds to probabilities for each topic being discussed during the corresponding interest rate decision meeting.

### 6.4. Topic models

Topic modeling is a method for classifying and mapping text into a given number of topics. Largely, topic modeling is a set of algorithms that automatically find and classify recurring patterns of words throughout a set of documents. In this paper, we use the LDA algorithm for the above task, which is a widely used algorithm in the field of computer science (Blei et al., 2003).

This algorithm assumes that each of the documents within our corpus consists of a mixture of *corpus-wide* topics. These topics, however, are not observable but are instead hidden behind everyday words and sentences. Specifically, LDA estimates what fraction of each document in a corpus can be assigned to each of the topics. The number of topics is set in advance. We do not observe the topics in the document, only the words that those topics tend to generate. LDA is an algorithm that employs an unsupervised learning approach, in that we do not set prior probabilities for any of the words belonging to any given topic. Besides, it is a mixed-membership model, and therefore the assumption is that every word in the corpus simultaneously belongs to several topics and the topic distributions vary over documents in the corpus.

To provide more intuition, consider an implicit assumption that a given set of words relates to a specific topic. For example, consider the following set of words: `gain`, `employment`, `labor`. Each of these words would map into an underlying topic "labor market" with a higher probability compared to what it would map into the topic of "economic growth". This algorithm has a considerable advantage, its objectivity. It makes it possible to find the best association between words and the underlying topics without preset word lists or labels. The LDA algorithm works its way up through the corpus. It first associates each word in the vocabulary to any given latent topic. It allows each word to have associations with multiple topics. Given these associations, it then proceeds to associate each document with topics. Besides the actual corpus, the main input that the model receives is how many topics there should be. Given those, the model will generate $\beta_k$ topic distributions, the distribution over words for each topic. The model will also generate $\theta_d$ document distributions for each topic, where $d$ is the number of documents. This modeling is done using Gibbs sampling iterations, going over each term in each document and assigning relative importance to each instance of the term.

In R, we use the package `topicmodels`, with the default parameter values supplied by the LDA function. Specifying a parameter is required before running the algorithm, which increases the subjectivity level. This parameter, $k$, is the number of topics that the algorithm should use to classify a given set of documents. There are analytical approaches to decide on the values of $k$, but most of the literature set it on an *ad hoc* basis. When choosing $k$ we have two goals that are in direct conflict with each other. We want to correctly predict the text and be as specific as possible to determine the number of topics. Yet, at the same time, we want to be able to interpret our results, and when we get too specific, the general meaning of each topic will be lost.

Let us demonstrate with this example by first setting $k = 2$, meaning that we assume only two topics to be present throughout our interest rate discussions. Below are the top seven words to be associated with these two topics. It can be seen below that while these two sets of words differ, they both have overlapping terms. This demonstrates that each word can be assigned to multiple topics but with a different probability.

Table 5 shows that Topic 1 relates directly and clearly to changes in the target rate, while Topic 2 relates more to inflationary expectations. However, these are not the only two things that the policymakers discuss during interest rate meetings, and we can safely assume that there should be more topics considered, meaning $k$ should be larger than two.[11]

[11] The supervised approach may help to determine the main theme of each topic objectively.

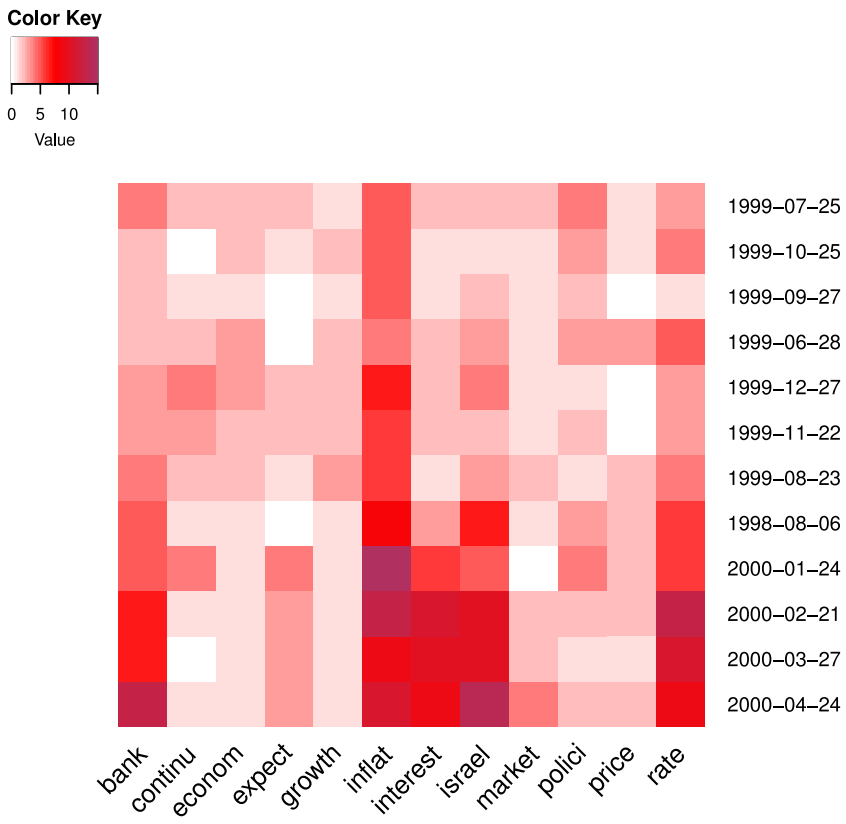


**Fig. 12.** Heatmap `tf` for 1999.
*Notes*: Heatmap for documents published in 1999 (`tf` weighted `dtm`). The color key corresponds to probabilities for each topic being discussed during the corresponding interest rate decision meeting.

**Table 5**
Topic-relevant Words - Topics 1 to 2.

| Topic 1 | Topic 2 |
|---|---|
| "increas" | "rate" |
| "rate" | "interest" |
| "month" | "expect" |
| "continu" | "israel" |
| "declin" | "inflat" |
| "discuss" | "bank" |
| "market" | "month" |
| "monetari" | "quarter" |

*Note*: Terms with the highest probability of appearing in Topic 1 and Topic 2.

**Table 6**
Topic-relevant Words - Topics 1 to 6.

| Topic 1 | Topic 2 | Topic 3 | Topic 4 | Topic 5 | Topic 6 |
|---|---|---|---|---|---|
| "declin" | "bank" | "increas" | "continu" | "quarter" | "interest" |
| "monetari" | "economi" | "month" | "rate" | "year" | "rate" |
| "discuss" | "month" | "interest" | "remain" | "rate" | "israel" |
| "rate" | "forecast" | "inflat" | "market" | "growth" | "inflat" |
| "data" | "market" | "hous" | "term" | "month" | "expect" |
| "polici" | "govern" | "continu" | "year" | "expect" | "discuss" |
| "indic" | "global" | "rate" | "price" | "first" | "bank" |
| "develop" | "activ" | "indic" | "growth" | "point" | "econom" |

*Note*: Terms with the highest probability of appearing in Topics 1 through 6.

To demonstrate the opposite side of the trade-off, let us consider $k = 6$, i.e., we assume six different topics are being discussed. Below is the top seven words with the highest probability to be associated with these six topics:

The division between topics is less clear in Table 6 compared to Table 5. While Topics 1, 2 and 3 relate to potential changes in interest rate, Topic 4 relates to housing market conditions, and Topic 5 relates to a higher level of expected growth taking into account monetary policy considerations. Topic 6 covers economic growth and banking discussions. We see that while we get more granularity in topics by increasing the possible number of topics, we see increased redundancy in the number of topics. Given this outcome, we could continue to adjust $k$ and assess the result.

Next, we demonstrate how we run this algorithm. First, we specify a set of parameters for Gibbs sampling. These include `burnin`, `iter`, `thin`, which are the parameters related to the amount of Gibbs sampling draws, and the way these are drawn.

```
burnin <- 4000
iter <- 2000
thin <- 500
seed <- list(2003, 5, 63, 100001, 765)
nstart <- 5
```

As discussed, the number of topics is decided arbitrarily, as an educated guess, and can be adjusted as needed. Here, based on the previous analysis we take the average of the two previous assumptions and end up with four assumed topics.

```
k <- 4
```

The code below runs the LDA algorithm, using the set of parameters as described above and our `dtm`:

```
lda.results <- LDA(dtm.sparse, k, method = "Gibbs",
control = list(nstart = nstart, seed = seed,
best = best, burnin = burnin, iter = iter, thin = thin))
```

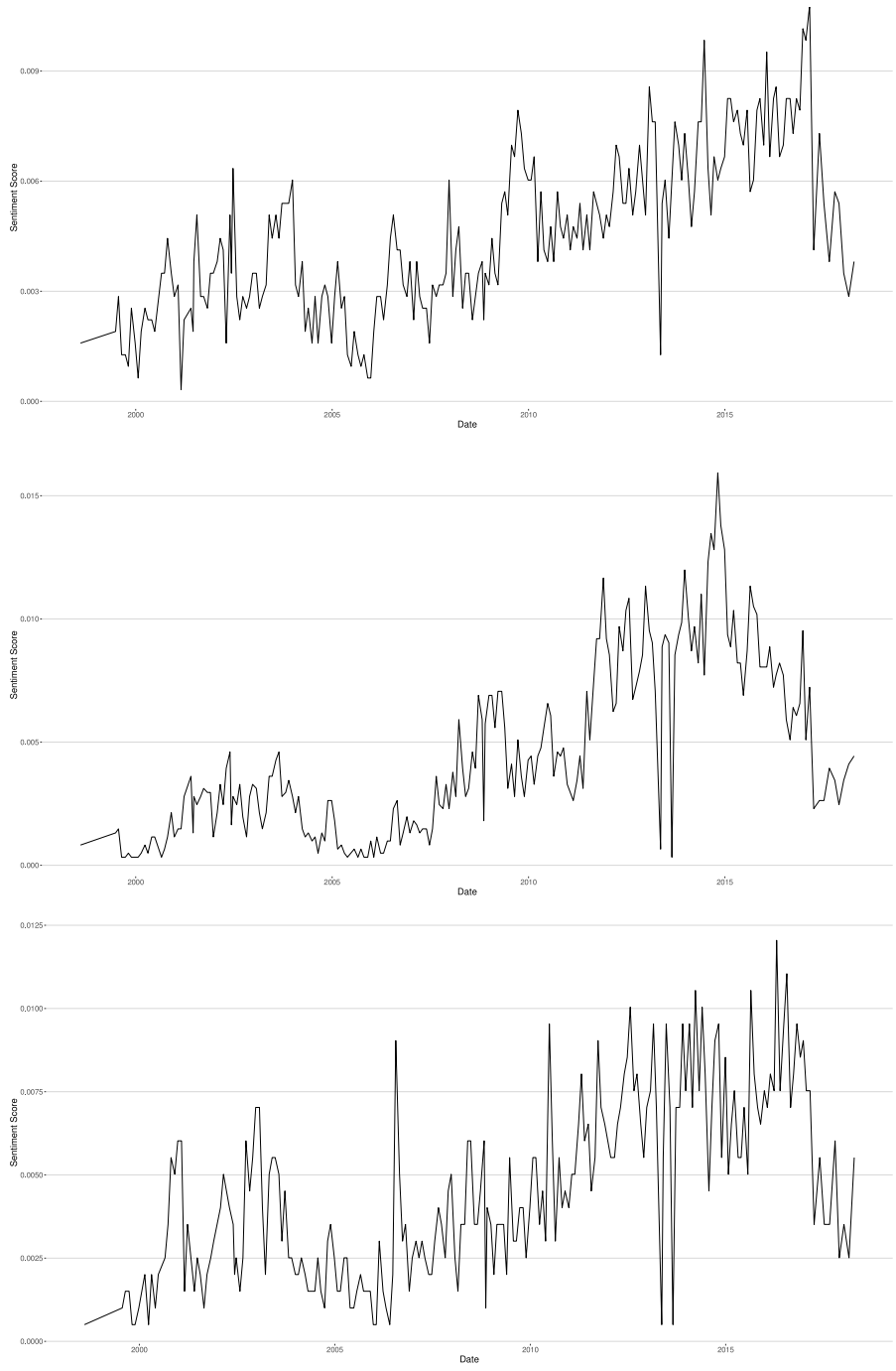


**Fig. 13.** Sentiment - Word Count.
*Notes*: Scaled count of positive (top panel), negative (middle panel), and uncertainty (bottom panel) words in each document using the dictionary approach.

These last lines write out and save the estimated topics as provided by the LDA algorithm:

```
lda.topics < - as.matrix(topics(lda.results))
lda.results.terms < -
as.matrix(terms (lda.results,8))
lda.results.terms
```

Let us examine the topics presented in Table 7. Topic 1 relates to current changes in interest rate and its goal of keeping inflation in range. It mentions the interest rate, inflation expectations and the range of inflation. Topic 2 relates to the actual inflation data and inflationary expectations. Topic 3 relates to a high-level monetary policy discussion, and Topic 4 relates to housing market conditions. LDA algorithm also generates probability distribution of topics over the corpus.

Fig. 16 is a heat map containing a sample for the period of June 2007 to April 2008. Given an assumption that only four topics are discussed during each interest rate meeting, the values presented in the legend are probabilities for each topic being discussed during the corresponding interest rate decision meeting.

For example, during the meeting of November 2008, the "Monetary Policy" topic was discussed with greater probability compared to the "Inflation" topic. As can be seen from Fig. 16, this occurrence stands out from the regular pattern.

Figs. 17 and 18 present the heat maps about the interest rate announcements during the year 2007 and 2000, respectively.

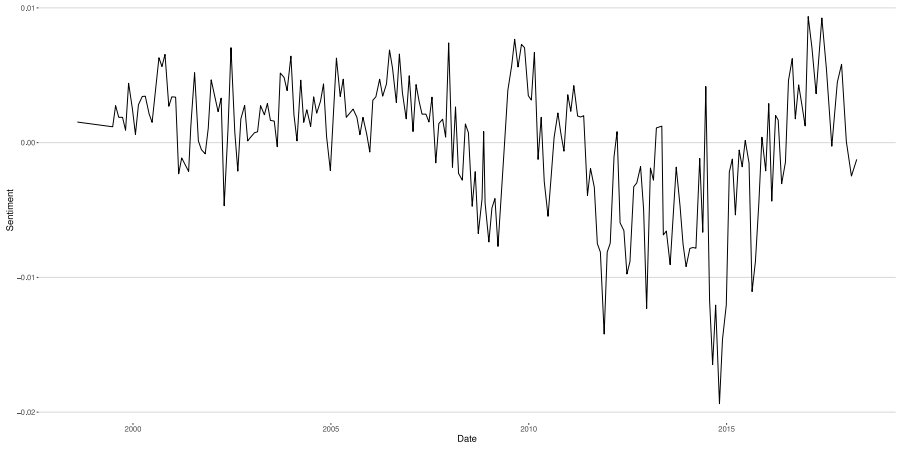


**Fig. 14.** Sentiment Indicator — Dictionary Approach.

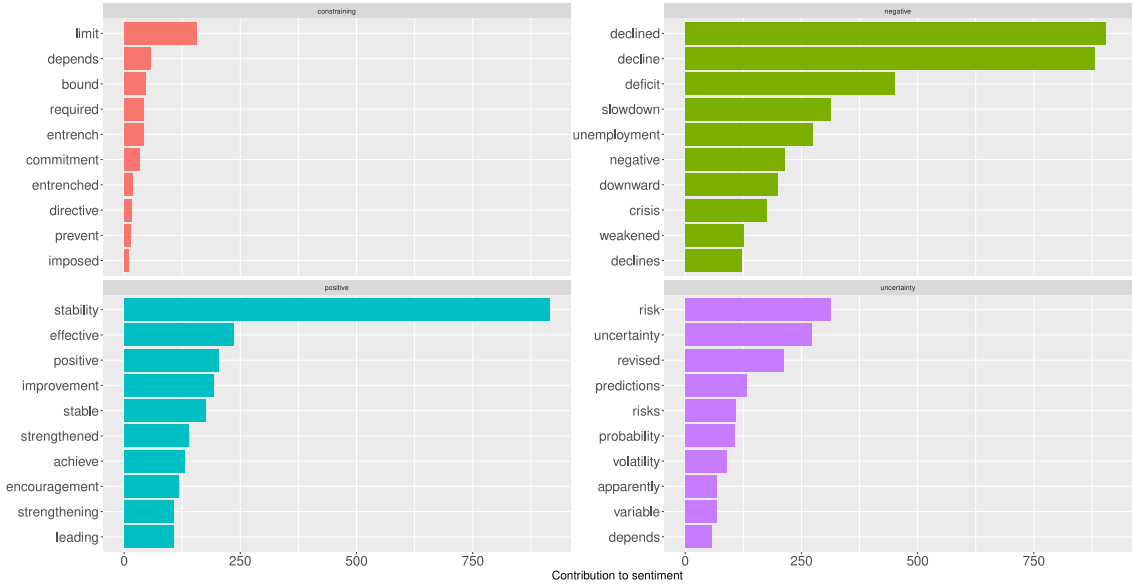


**Fig. 15.** Sentiment Contributions.
*Note*: This figure represents how much each term contributes to the sentiment in each corresponding category. These categories are defined as mutually exclusive. Constraining (top left), positive (top right), negative (bottom left), and uncertainty (bottom right) sentiments are represented.

Fig. 17 shows that in a given set of documents, the bulk of the discussion was spent on discussing the key interest rate set by the Bank of Israel. In contrast, it can be seen that inflation was not discussed at all during certain periods. Fig. 18 shows that the subject of discussion was mainly monetary policy during this period of time.

The remaining question is how to determine whether the topics selected by the LDA algorithm are a good fit for the underlying text. For that, we can use topic coherence. Topic coherence allows for a set of words, as generated by a topic model, to be rated for coherence or interpretability, as shown in Newman et al. (2010). One way to implement this in R is to use the `topicdoc` package, based on the paper by Boyd-Graber et al. (2014). This package can be applied on top of the analysis described in the previous section, and contains many helpful functions that allow for diagnostics of the estimated LDA model.

One can run a full set of diagnostics using `topic_diagnostics`, a function that takes the fitted LDA model as an input, and produces a set of measures that evaluate the fit, such as mean token length, exclusivity, coherence, and document prominence of each topic.

Another way to explore and evaluate the output of an LDA model is to visualize it. In R, this can be done with the help of `LDAvis` package (Mabey, 2018; Sievert & Shirley, 2014). LDA visualization can help answer questions such as what is the meaning of each topic; how prevalent is each topic; and how the estimated topics relate to each other. Together with the `shiny` package, the `LDAvis` provides an interactive tool that can be used to create visualizations such as the intertopic distance map.

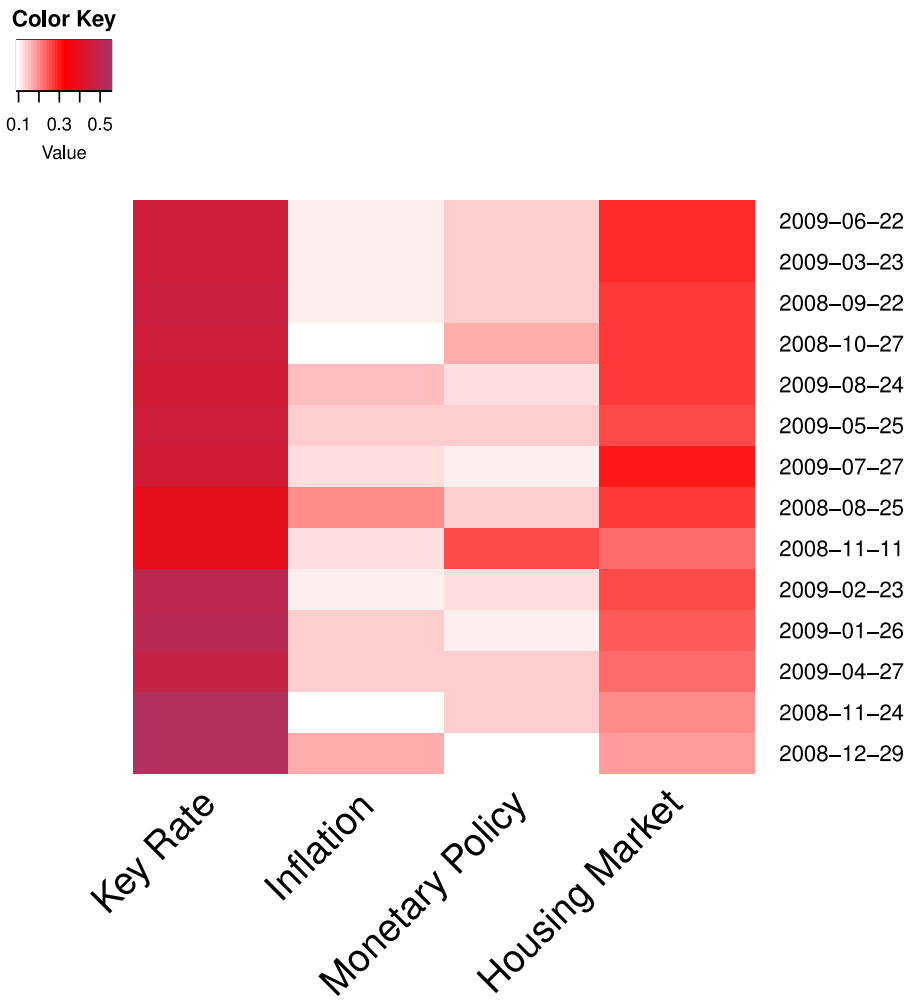


**Fig. 16.** Probability Distribution — Topics 1 to 4 from 2007 to 2008.
*Note*: The color key corresponds to probabilities for each topic being discussed during the corresponding interest rate decision meeting.

**Table 7**
Topic-relevant Words — Topics 1 to 4.

| Topic 1 | Topic 2 | Topic 3 | Topic 4 |
|---|---|---|---|
| "expect" | "increas" | "interest" | "month" |
| "continu" | "declin" | "rate" | "increas" |
| "rate" | "continu" | "stabil" | "rate" |
| "inflat" | "rate" | "israel" | "forecast" |
| "interest" | "expect" | "bank" | "bank" |
| "rang" | "remain" | "inflat" | "indic" |
| "israel" | "growth" | "market" | "growth" |
| "last" | "term" | "govern" | "year" |
| "price" | "nis" | "year" | "previous" |
| "bank" | "year" | "target" | "index" |
| "econom" | "data" | "term" | "hous" |

*Note*: Words with the highest probability of appearing in Topics 1 through 4.

The intertopic distance map is a visualization of the topics in a two-dimensional space. The area of these topic circles is proportional to the number of words belonging to each topic across the corpus. The circles are plotted using a multidimensional scaling algorithm based on the words they comprise, so topics that are closer together have more words in common. This algorithm also considers the saliency of terms, identifying the most informative or useful words for identifying topics within the corpus. Higher saliency values indicate that a word is more helpful in identifying a specific topic than a randomly selected term.

## 7. Conclusion

In this paper, we review some of the most commonly used text mining methodologies. We demonstrate how text sentiment and topics can be extracted from a set of text sources. Taking advantage of the open-source software package R and related packages, we provide a detailed step-by-step tutorial, including code excerpts that are easy to implement and examples of output. The framework we demonstrate in this paper shows how to process and utilize text data in an objective and automated way.

As described, the ultimate goal of this tutorial is to show how text analysis can help uncover the information embedded in monetary policy communication and to be able to organize it consistently. We first show how to set up a directory and input a set of relevant files into R. We show how to represent and store this set of files as a *corpus*, allowing for easy text manipulations. We then describe a series of text cleaning procedures that sets the stage for further text analysis. In the second part of the paper, we demonstrate approaches to initial text analysis, showing how to create several summary statistics for our existing corpus. We then describe two common approaches to text sentiment extraction and one common approach to topic modeling.

We also consider term-weighting and contiguous sequence of words (n-grams) to better capture the subtlety of central bank communication. We consider field-specific weighted lexicons, consisting of two, three, or four-word clusters, relating to a specific policy term being discussed. This approach provides an alternative and sometimes more precise picture of the text content, as opposed to individual terms, and allows us to find underlying patterns and linkages within text more precisely.

NLP methods have come a long way in the past few decades. These methods continue to evolve rapidly, getting increasingly more adept at capturing and understanding the nuances and context of language. At the same time, the ability to capture signals such as sentiment, topics, and semantic relationships in the text remains an important and relevant building block of language processing.

## CRediT authorship contribution statement

**Jonathan Benchimol:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Supervision. **Sophia Kazinnik:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Yossi Saadon:** Formal analysis, Investigation, Resources, Data curation, Writing – review & editing, Project administration, Funding acquisition.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

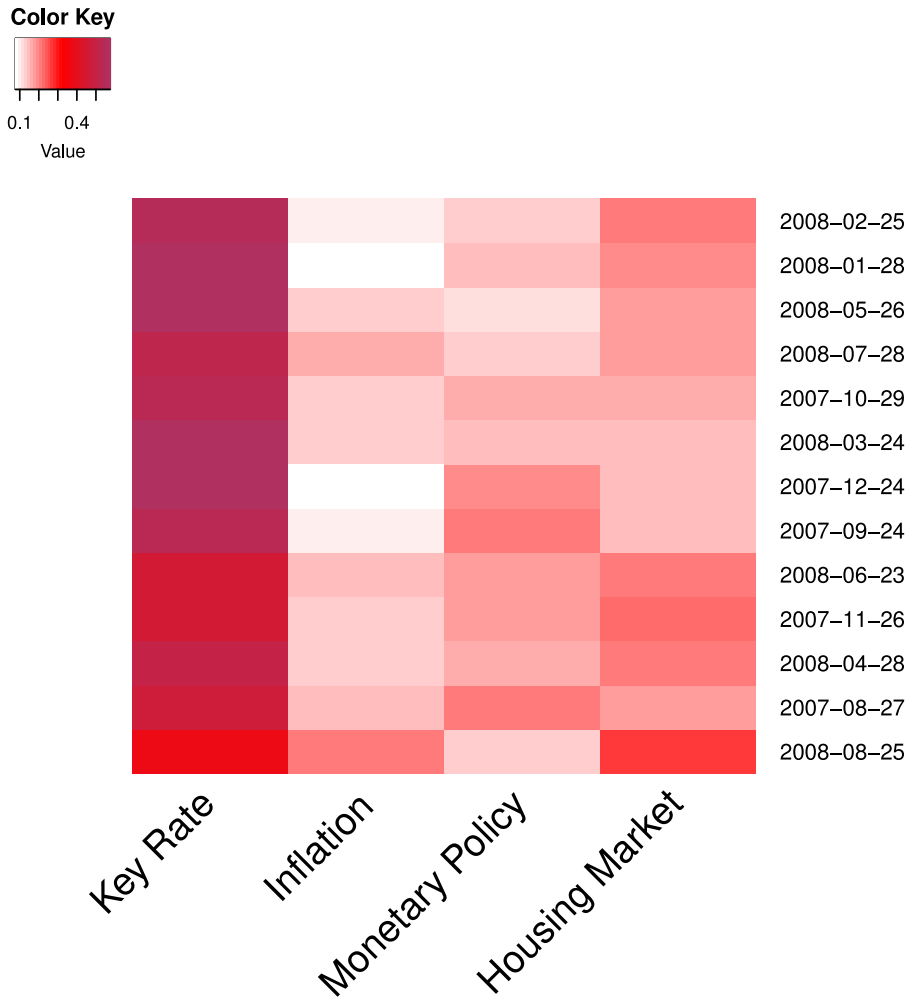


**Fig. 17.** Probability Distribution — Topics 1 to 4 from 2008 to 2009.
*Note*: The color key corresponds to probabilities for each topic being discussed during the corresponding interest rate decision meeting.

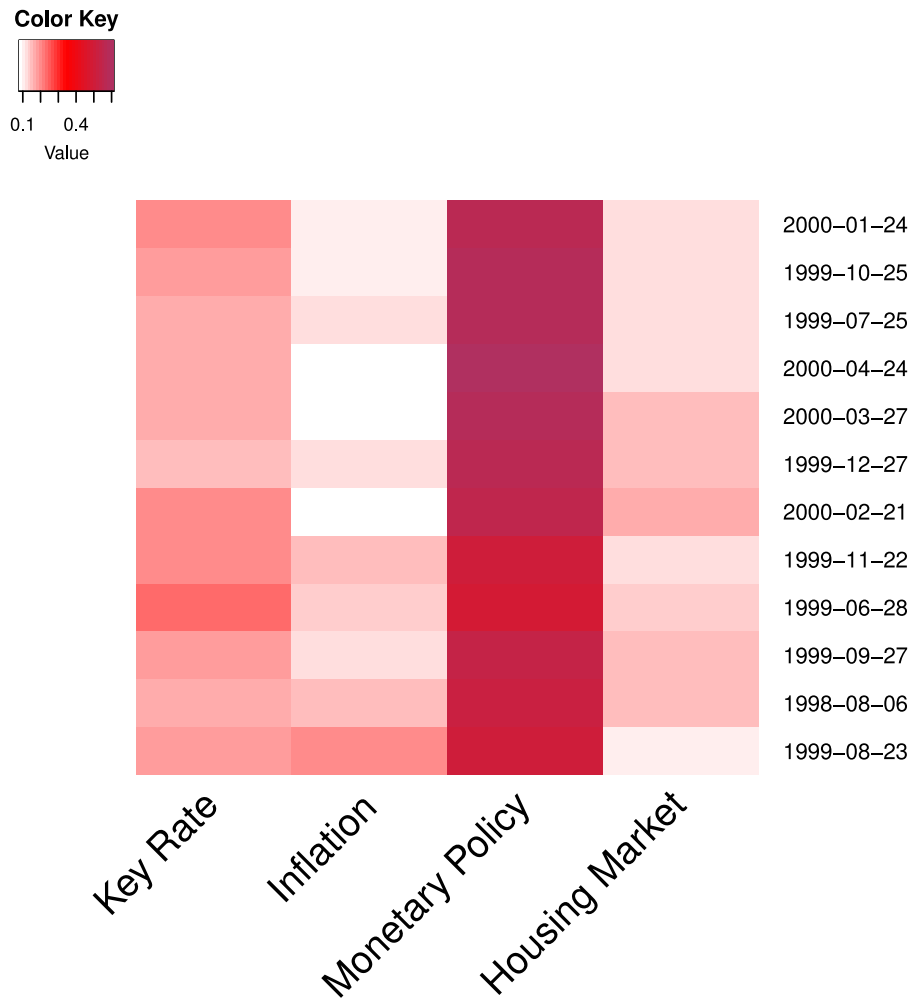


**Fig. 18.** Probability Distribution — Topics 1 through 4 from 1999 to 2000.
*Note*: The color key corresponds to probabilities for each topic being discussed during the corresponding interest rate decision meeting.

## References


Benchimol, J., Kazinnik, S., & Saadon, Y. (2020). Communication and transparency through central bank texts. In *132nd annual meeting of the American Economic Association*.

Benchimol, J., Kazinnik, S., & Saadon, Y. (2021). Federal reserve communication and the COVID-19 pandemic. *Covid Economics*, *79*, 218–256.

Bholat, D., Broughton, N., Ter Meer, J., & Walczak, E. (2019). Enhancing central bank communications using simple and relatable information. *Journal of Monetary Economics*, *108*(C), 1–15.

Bholat, D., Hans, S., Santos, P., & Schonhardt-Bailey, C. (2015). Text mining for central banks. In *Handbooks*, (33), Centre for Central Banking Studies, Bank of England.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Boyd-Graber, J., Mimno, D., & Newman, D. (2014). Care and feeding of topic models: Problems, diagnostics, and improvements. In E. M. Airoldi, D. Blei, E. A. Erosheva, & S. E. Fienberg (Eds.), *CRC Handbooks of Modern Statistical Methods*, *Handbook of Mixed Membership Models and their Applications* (pp. 225–254). Boca Raton, Florida: CRC Press.

Bruno, G. (2017). Central bank communications: information extraction and semantic analysis. In Big Data (Ed.), *IFC bulletins chapters*: *vol. 44*, *Bank for international settlements* (pp. 1–19). Bank for International Settlements.

Calomiris, C. W., & Mamaysky, H. (2020). How natural language processing will improve central bank accountability and policy. *Cato Journal*, *40*(2), 447–465.

Carley, K. (1993). Coding choices for textual analysis: A comparison of content analysis and map analysis. *Sociological Methodology*, *23*, 75–126.

Correa, R., Garud, K., Londono, J. M., & Mislang, N. (2021). Sentiment in central banks' financial stability reports. *Review of Finance*, *25*(1), 85–120.

Danowski, J. A. (1993). Network analysis of message content. *Progress in Communication Sciences*, *12*, 198–221.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407.

Diesner, J. (2013). From texts to networks: detecting and managing the impact of methodological choices for extracting network data from text data. *KI - Künstliche Intelligenz*, *27*(1), 75–78.

Diesner, J., & Carley, K. M. (2011a). Semantic networks. In G. B. J. Golson (Ed.), *Encyclopedia of social networking*, (pp. 766–769). Thousand Oaks, CA: Sage Publications.

Diesner, J., & Carley, K. M. (2011b). Words and networks. In G. B. J. Golson (Ed.), *Encyclopedia of social networking* (pp. 958–961). Thousand Oaks, CA: Sage Publications.

Ehrmann, M., & Fratzscher, M. (2007). Communication by central bank committee members: Different strategies, same effectiveness? *Journal of Money, Credit and Banking*, *39*(2–3), 509–541.

Feinerer, I., Hornik, K., & Meyer, D. (2008). Text mining infrastructure in R. *Journal of Statistical Software*, *25*(i05).

Hansen, S., & McMahon, M. (2016). Shocking language: Understanding the macroeconomic effects of central bank communication. *Journal of International Economics*, *99*(S1), 114–133.

Hansen, S., McMahon, M., & Tong, M. (2019). The long-run information effect of central bank communication. *Journal of Monetary Economics*, *108*(C), 185–202.

Laver, M., Benoit, K., & Garry, J. (2003). Extracting policy positions from political texts using words as data. *American Political Science Review*, *97*(02), 311–331.

Loughran, T., & Mcdonald, B. (2011). When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *The Journal of Finance*, *66*(1), 35–65.

Lucca, D. O., & Trebbi, F. (2009). Measuring central bank communication: an automated approach with application to FOMC statements. NBER Working Papers 15367, National Bureau of Economic Research.

Mabey, B. (2018). pyLDAvis: Python library for interactive topic model visualization. Port of the R package LDAvis.

Newman, D., Lau, J. H., Grieser, K., & Baldwin, T. (2010). Automatic evaluation of topic coherence. In *Human language technologies: the 2010 annual conference of the north American chapter of the association for computational linguistics* (pp. 100–108). Association for Computational Linguistics.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

Sievert, C., & Shirley, K. (2014). LDAvis: a method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces* (pp. 63–70). Baltimore, Maryland, USA: Association for Computational Linguistics.

Ter Ellen, S., Larsen, V. H., & Thorsrud, L. A. (2022). Narrative monetary policy surprises and the media. *Journal of Money, Credit and Banking* (in press).

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, *58*(301), 236–244.

Wickham, H. (2014). Tidy data. *Journal of Statistical Software*, *59*(i10), 1–23.