BigSEM for Text Data

Text data is increasingly recognized as a rich source of information, offering insights that traditional quantitative measures may overlook. Modern natural language processing (NLP) offers a variety of techniques for analyzing text, such as sentiment analysis (Wankhade et al., 2022), topic modeling (Vayansky & Kumar, 2020), and word embedding (Wang et al., 2019). These techniques automatically extract information from text and transform it into meaningful values or vectors, by-passing the need for labor-intensive manual coding.

Structural equation modeling (SEM) is a popular tool in the social and behavioral sciences for analyzing relationships between observed and latent variables. Incorporating textual data into SEM provides a promising avenue for researchers to integrate qualitative and quantitative data analysis. In response to this opportunity, we developed TextSEM, an R package designed to incorporate text data within SEM frameworks. This package leverages advanced NLP techniques to convert text into latent variables, integrate them into SEM model, and conduct estimation.

Here, we demonstrate the practical application of TextSEM through examples using a teaching evaluation dataset.

Example data

For illustration, we use a set of student evaluation of teaching data. The data were scraped from an online website conforming to its site requirement, containing 38,240 teaching evaluations on 1,000 instructors.

For each evaluation, we have information on the overall numerical rating of the teaching of the instructor, how difficult the class was, whether the student took the class for credit or not, grade the student received, etc. The data also contain short textual comments about the instructor's teaching, as well as a list of tabs describing the course. Part of the data are shown below:

```
'data.frame' : 38240 obs. of  13 variables:
 $ id        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ profid    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ rating    : num  5 5 4 3 1 5 5 2 3 3 ...
 $ difficulty: int  3 4 5 5 5 5 5 4 5 5 ...
 $ credit    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ grade     : int  5 4 5 7 3 NA 6 7 7 8 ...
 $ book      : int  0 0 0 0 0 1 1 1 1 1 ...
 $ take      : int  1 1 1 0 0 0 1 0 NA NA ...
 $ attendance: int  1 1 0 1 1 1 1 1 1 0 ...
 $ tags      : chr  "respected;accessible outside class;skip
                 class? you won't pass ." "accessible outside
                 class;lots of homework;respected" "tough
                 grader;lots of homework;accessible outside
                 class" "tough grader;so many papers;lots of
                 homework" ...
 $ comments  : chr  "best professor i've had in college . only
                 thing i dont like is the writing assignments"
                 "Professor has been the best math professor
                 I've had at thus far . He assigns a heavy
                 amount of homework but "| __truncated__ "He
                 was a great professor . he does give a lot
                 of homework but he will work with you if you
                 don't clearly unders"| __truncated__
                 "Professor is an incredibly respected teacher,
                 however his class is extremely difficult . I
                 believe he just ass"| __truncated__ ...
```

```
$ date      : chr  "04/17/2018" "02/13/2018" "01/07/2018"
                "12/11/2017" ...
$ gender    : num  1 1 1 1 1 1 1 1 1 1 ...
```

Text Sentiment

Sentiment analysis is the process of systematically identifying and quantifying the sentiment expressed in a text.

Lexicon-based / dictionary-based approach

A common method is the lexicon-based approach, where each word is assigned a sentiment score, and the overall sentiment of a sentence is calculated as a weighted average of the words within it. Here, we adopt the approach used by `sentimentr` (Rinker, 2017), which utilizes a lexicon of polarized words (Hu & Liu, 2004; Jockers, 2017) and adjusts these scores with valence shifters.

The lexicon-based sentiment analysis begins with tokenization, where each paragraph (p_i) is broken down into individual sentences ($s_{1}, s_{2}, \cdots,s_{n}$), and each sentence ($s_{j}$) is further decomposed into a sequence of words (${w_{1}, w_{2}, \cdots,w_{m}}$). Thus, each word can be represented as $w_{i, j, k}$. For instance, $w_{2,3,1}$ refers to the first word in the third sentence of the second paragraph.

Next, the words $w_{i, j, k}$ in each sentence are compared against a dictionary of polarized words. Positive words $(w_{i, j, k}^+)$ and negative words $(w_{i, j, k}^-)$ are assigned scores of +1 and -1, respectively. The context surrounding each polarized word is then analyzed, identifying neutral words $(w_{i, j, k}^0)$, negative modifiers $(w_{i, j, k}^n)$, amplifiers $(w_{i, j, k}^a)$, and de-amplifiers $(w_{i, j, k}^d)$. The sentiment score of each word is first weighted by its own score, and then further adjusted based on the function and quantity of valence shifters within its context. The sentiment score of the text is the average sentiment score of all words in the text.

AI-based sentiment analysis

The Korn Ferry Institute's AITMI team made sentiment.ai for researchers and tinkerers who want a straight-forward way to use powerful, open source deep learning models to improve their sentiment analyses. Wiseman et al. (2022) packed the method in an R package `sentiment.ai` that can produce the sentiment of text and it outperforms many other methods.

The method is based on the Universal Sentence Embedding that embeds a text into a 512 by 1 vector. Then, it build a model between the embedded vector and the labels between the text for prediction.

Online app

We have developed online apps for both dictionary-based and AI-based sentiment analysis. We created a video to show how to use the AI-based methods to get the sentiment of a text variables. The obtained sentiment score is saved as a new variable in the data set that can be used in further data analysis.

Text Embedding and Encoders

Embedding techniques are widely used in modern NLP. These methods transform text into numerical vectors, capturing both semantic and syntactic relationships with high fidelity (Patil et al., 2023). Conceptually, this process can be viewed as factor analysis or principal component analysis of the text to extract latent information. However, compared to those techniques, embedding vectors are usually of higher dimensionality (e.g., 768 dimensions), which allows for a more detailed representation of semantic and linguistic features.

The evolution of word embedding techniques has been substantial, from basic one-hot encoding to approaches such as Word2Vec, GloVe, and transformer-based models. Notably, transformer models like BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) and SentenceBERT (Reimers & Gurevych, 2019) have significantly advanced context-aware sentence embeddings. These models are initially pre-trained on extensive text corpora and can be fine-tuned for specific applications, enhancing their adaptability and effectiveness. BERT utilizes a deep bidirectional transformer architecture to produce contextualized word embeddings that are aggregated into sentence representations. SentenceBERT modifies BERT to optimize it for sentence-level tasks by fine-tuning with natural language inference data, which enhances the ability to compare sentence embeddings via cosine similarity. This optimization boosts BERT's efficiency and effectiveness in applications such as semantic similarity assessment and information retrieval.

Furthermore, the development of Large Language Models (LLMs) has improved text embedding generation. OpenAI, for instance, offers several GPT-based embedding models through its API services, including the "text-embedding-3-small" and the more robust "text-embedding-3-large" model (OpenAI, 2024). These models have demonstrated great capabilities across a diverse set of tasks, including semantic search, clustering, and recommendation systems.

TextSEM supports the integration of both SentenceBERT models and OpenAI APIs for generating text embeddings. However, the high dimensionality of these embeddings poses challenges for direct SEM model estimation. To mitigate this, TextSEM employs Principal Component Analysis (PCA) to reduce dimensionality, allowing users to tailor the reduced dimensions to their specific requirements.

Our online app can directly embed text into vectors and save the vectors as an R data set.

AI based text sentiment

Analysis Menu

List of variables		Text variable
id profid rating difficulty credit grade book take attendance tags	→ ←	comments

Options

Embedding model all-mpnet-base-v2

RUN

Note that the analysis may take a while to complete. Please be patient and do not refresh the page.

Use of the R package TextSEM

The R package TextSEM can be used for SEM analysis with text data. To install the package, please use

```
## Install the package for text analysis
remotes::install_github("Stan7s/TextSEM")

## The package can be installed from CRAN directly in the future
# install.packages('TextSEM')
```

We now illustrate the use of the package through several examples.

Sentiment analysis

In this example, we introduce how to use the function `sem.sentiment` to extract sentiment variables from text and estimate the SEM model. Specifically, the overall sentiment of comment is extracted and used as a mediator between three endogenous variables (book, attendance, difficulty) and two exogenous variables (grade and rating).

To use this function, we need to first specify the model:

```
model <- ' rating ~ book + attendance + difficulty + comments
           grade ~ book + attendance + difficulty + comments
           comments ~ book + attendance + difficulty
         '
```

The function `sem.sentiment` requires three parameters: the structural equation model, the input data frame, and the name of the text variable in the data frame to be analyzed for sentiment.

```
res <- sem.sentiment(model = model,
                     data = prof1000,
                     text_var=c('comments'))
summary(res$estimates, fit = TRUE)
```

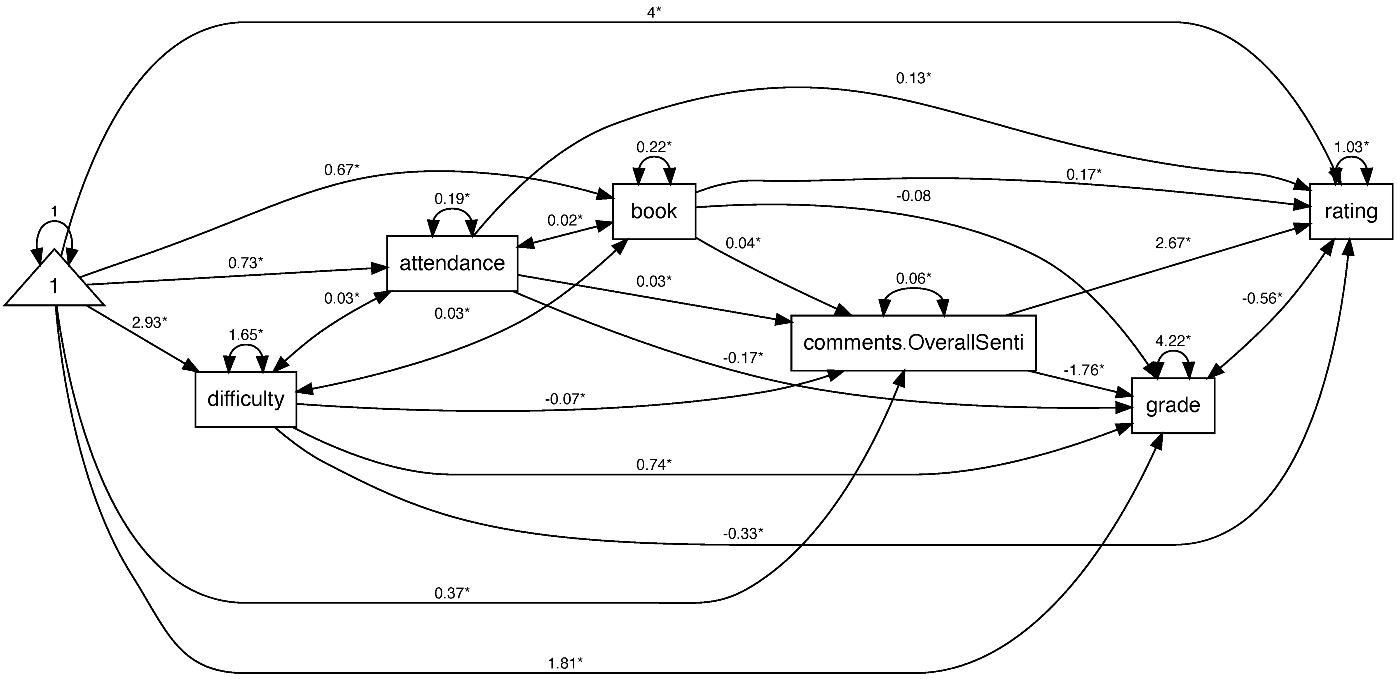
The output of the analysis is given below:

```
lavaan 0.6.17 ended normally after 63 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        27

  Number of observations                         38240
  Number of missing patterns                         8

Model Test User Model:

  Test statistic                                 0.000
  Degrees of freedom                                 0

Model Test Baseline Model:

  Test statistic                             31563.154
  Degrees of freedom                                12
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    1.000
  Tucker-Lewis Index (TLI)                       1.000

  Robust Comparative Fit Index (CFI)             1.000
  Robust Tucker-Lewis Index (TLI)                1.000

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)            -160948.572
  Loglikelihood unrestricted model (H1)    -160948.572

  Akaike (AIC)                              321951.144
  Bayesian (BIC)                            322182.038
  Sample-size adjusted Bayesian (SABIC)     322096.232

Root Mean Square Error of Approximation:
```

```
  RMSEA                                          0.000
  90 Percent confidence interval - lower         0.000
  90 Percent confidence interval - upper         0.000
  P-value H_0: RMSEA <= 0.050                       NA
  P-value H_0: RMSEA >= 0.080                       NA

  Robust RMSEA                                   0.000
  90 Percent confidence interval - lower         0.000
  90 Percent confidence interval - upper         0.000
  P-value H_0: Robust RMSEA <= 0.050                NA
  P-value H_0: Robust RMSEA >= 0.080                NA

Standardized Root Mean Square Residual:

  SRMR                                           0.000

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Observed
  Observed information based on                Hessian

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  rating ~
    book              0.169    0.013   12.905    0.000
    attendance        0.127    0.023    5.618    0.000
    difficulty       -0.331    0.004  -75.262    0.000
    cmmnts.OvrllSn    2.671    0.021  125.974    0.000
  grade ~
    book             -0.080    0.051   -1.558    0.119
    attendance       -0.170    0.056   -3.058    0.002
    difficulty        0.742    0.020   36.382    0.000
    cmmnts.OvrllSn   -1.756    0.102  -17.171    0.000
  comments.OverallSenti ~
    book              0.043    0.003   13.053    0.000
    attendance        0.031    0.006    5.290    0.000
    difficulty       -0.074    0.001  -73.666    0.000
```

```
Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  .rating ~~
    .grade           -0.558    0.024  -23.191    0.000
  book ~~
    attendance        0.017    0.002    8.374    0.000
    difficulty        0.030    0.004    8.650    0.000
  attendance ~~
    difficulty        0.028    0.006    4.712    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .rating            3.995    0.022  182.815    0.000
   .grade             1.807    0.084   21.559    0.000
   .cmmnts.OvrllSn    0.367    0.005   69.455    0.000
    book              0.673    0.003  245.275    0.000
    attendance        0.732    0.004  164.946    0.000
    difficulty        2.928    0.007  445.625    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .rating            1.034    0.008  136.885    0.000
   .grade             4.222    0.069   61.443    0.000
   .cmmnts.OvrllSn    0.061    0.000  136.529    0.000
    book              0.220    0.002  120.633    0.000
    attendance        0.195    0.003   71.124    0.000
    difficulty        1.651    0.012  138.275    0.000
```

The path diagram for the model is



Topic modeling

Students' comments about an instructor typically cover multiple topics, such as teaching style, classroom climate, and homework assignments. To identify these topics exploratorily and understand their relationships with other variables, we can apply the `sem.topic` function. This function performs topic modeling and estimates the SEM model including those identified topics.

In this example, we combine the comments from multiple students for each instructor. We also get the average scores for other variables.

```
prof.nest <- prof1000 %>% group_by(profid) %>%
summarise(comments = paste(comments, collapse = " "),
        tags = paste(tags, collapse = ";"),
        rating = mean(rating, na.rm = TRUE),
        difficulty=mean(difficulty, na.rm = TRUE),
        book = mean(book, na.rm = TRUE),
        grade=mean(grade, na.rm = TRUE))
```

In addition to the three required parameters for `sem.sentiment` – model, data, and text variables, the `sem.topic` function requires an additional parameter: `n_topics`. This parameter specifies the number of topics to extract from each column of the text data. Based on previous cross-validation analysis (Jacobucci et al., 2023), six topics were identified in this dataset. Consequently, we will extract six topics. Note that only the first n − 1 topics will be incorporated into the SEM to avoid perfect multicollinearity, where n is the total number of topics specified.

```
model <- ' rating ~ book + difficulty + comments'
res <- sem.topic(model = model,
                 data = prof.nest,
                 text_var = c('comments'),
                 n_topics = c(6))
summary(res$estimates, fit=TRUE)
```

The output is given below:

```
lavaan 0.6.17 ended normally after 1 iteration

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                         8

                                                  Used       Total
  Number of observations                           984        1000

Model Test User Model:

  Test statistic                                 0.000
  Degrees of freedom                                 0

Model Test Baseline Model:

  Test statistic                              1143.062
  Degrees of freedom                                 7
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    1.000
  Tucker-Lewis Index (TLI)                       1.000

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)               -631.624
  Loglikelihood unrestricted model (H1)       -631.624

  Akaike (AIC)                                1279.248
```

```
  Bayesian (BIC)                              1318.381
  Sample-size adjusted Bayesian (SABIC)       1292.973

Root Mean Square Error of Approximation:

  RMSEA                                          0.000
  90 Percent confidence interval - lower         0.000
  90 Percent confidence interval - upper         0.000
  P-value H_0: RMSEA <= 0.050                       NA
  P-value H_0: RMSEA >= 0.080                       NA

Standardized Root Mean Square Residual:

  SRMR                                           0.000

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  rating ~
    book              0.295    0.058    5.094    0.000
    difficulty       -0.335    0.023  -14.663    0.000
    comments.topc1    0.392    0.106    3.696    0.000
    comments.topc2    2.503    0.102   24.531    0.000
    comments.topc3    1.637    0.105   15.554    0.000
    comments.topc4   -0.344    0.090   -3.799    0.000
    comments.topc5    0.273    0.093    2.955    0.003

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .rating            0.211    0.010   22.181    0.000
```

Text embedding

Embedding techniques offer an advantage over topic models in their ability to construct latent factors in higher dimensions from textual data. In this example, we demonstrate how to leverage embedding techniques within the framework of SEM using the `sem.emb` function.

Before we start, we need to set up the Python environment with the `reticulate` package, which provides a bridge between R and Python. The code below can be used for the purpose.

```
library(reticulate)


## First time set-up
virtualenv_create("r-reticulate")
py_install("transformers")
py_install("torch")
py_install("sentence_transformers")
py_install("openai")


## Call virtual environment
use_virtualenv("r-reticulate")
```

Although it is not required, we recommended first to embed the text and then include the embedded vectors in the SEM analysis .The reason is that text embedding can be time consuming. The embedded data can also be used in multiple models rather than just the model specified.

We can use the `sem.encode` function to generate text embeddings. This function supports pre-trained models from SentenceBERT and OpenAI. Here, we'll use the all-mpnet-base-v2 model from SentenceBERT. Note that when using OpenAI models, an API key must be specified in the system directory.

```
embeddings <- sem.encode(prof.nest$comments,
                         encoder = "all-mpnet-base-v2")
## save the embeddings
save(embeddings, file="data/prof.nest.emb.rda")
```

We then incorporate these embeddings into an SEM model using the `sem.emb` function. This function allows us to integrate the rich semantic information captured by the embeddings into our statistical model. Two key parameters in this function are: 1) `pca_dim`: the number of dimensions to retain after applying PCA to the embeddings, and 2) `emb_filepath`: the file path to the saved embeddings.

```
sem_model <- ' rating ~ book + difficulty + comments'
res <- sem.emb(sem_model = sem_model,
               data = prof.nest,
               text_var = "comments",
```

```
            pca_dim = 10,
            emb_filepath = "data/prof.nest.emb.rda")
```

The output looks like:

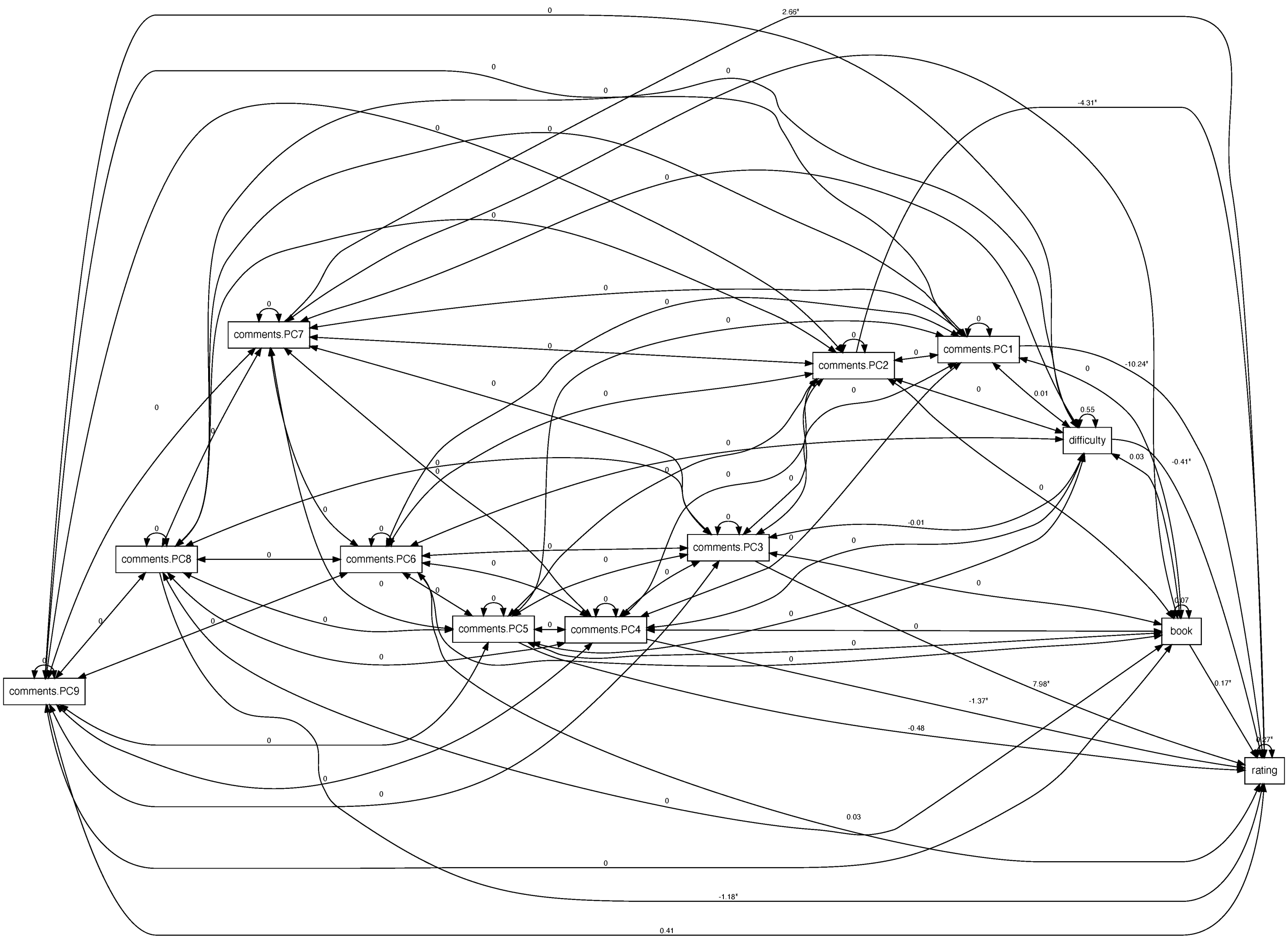
```
lavaan 0.6.17 ended normally after 1 iteration

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        12

                                                  Used       Total
  Number of observations                           984        1000

Model Test User Model:

  Test statistic                                 0.000
  Degrees of freedom                                 0

Model Test Baseline Model:

  Test statistic                               887.411
  Degrees of freedom                                11
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    1.000
  Tucker-Lewis Index (TLI)                       1.000

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)               -759.449
  Loglikelihood unrestricted model (H1)       -759.449

  Akaike (AIC)                                1542.898
  Bayesian (BIC)                              1601.598
  Sample-size adjusted Bayesian (SABIC)       1563.486

Root Mean Square Error of Approximation:
```

```
  RMSEA                                          0.000
  90 Percent confidence interval - lower         0.000
  90 Percent confidence interval - upper         0.000
  P-value H_0: RMSEA <= 0.050                       NA
  P-value H_0: RMSEA >= 0.080                       NA

Standardized Root Mean Square Residual:

  SRMR                                           0.000

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  rating ~
    book              0.168    0.067    2.517    0.012
    difficulty       -0.406    0.026  -15.524    0.000
    comments.PC1    -10.239    0.549  -18.654    0.000
    comments.PC2     -4.308    0.539   -7.998    0.000
    comments.PC3      7.982    0.573   13.931    0.000
    comments.PC4     -1.373    0.526   -2.612    0.009
    comments.PC5     -0.484    0.534   -0.906    0.365
    comments.PC6      0.034    0.531    0.064    0.949
    comments.PC7      2.664    0.531    5.019    0.000
    comments.PC8     -1.183    0.527   -2.243    0.025
    comments.PC9      0.408    0.531    0.767    0.443

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .rating            0.274    0.012   22.181    0.000
```

Note that to embed the text and conduct the analysis at the same time, one can use

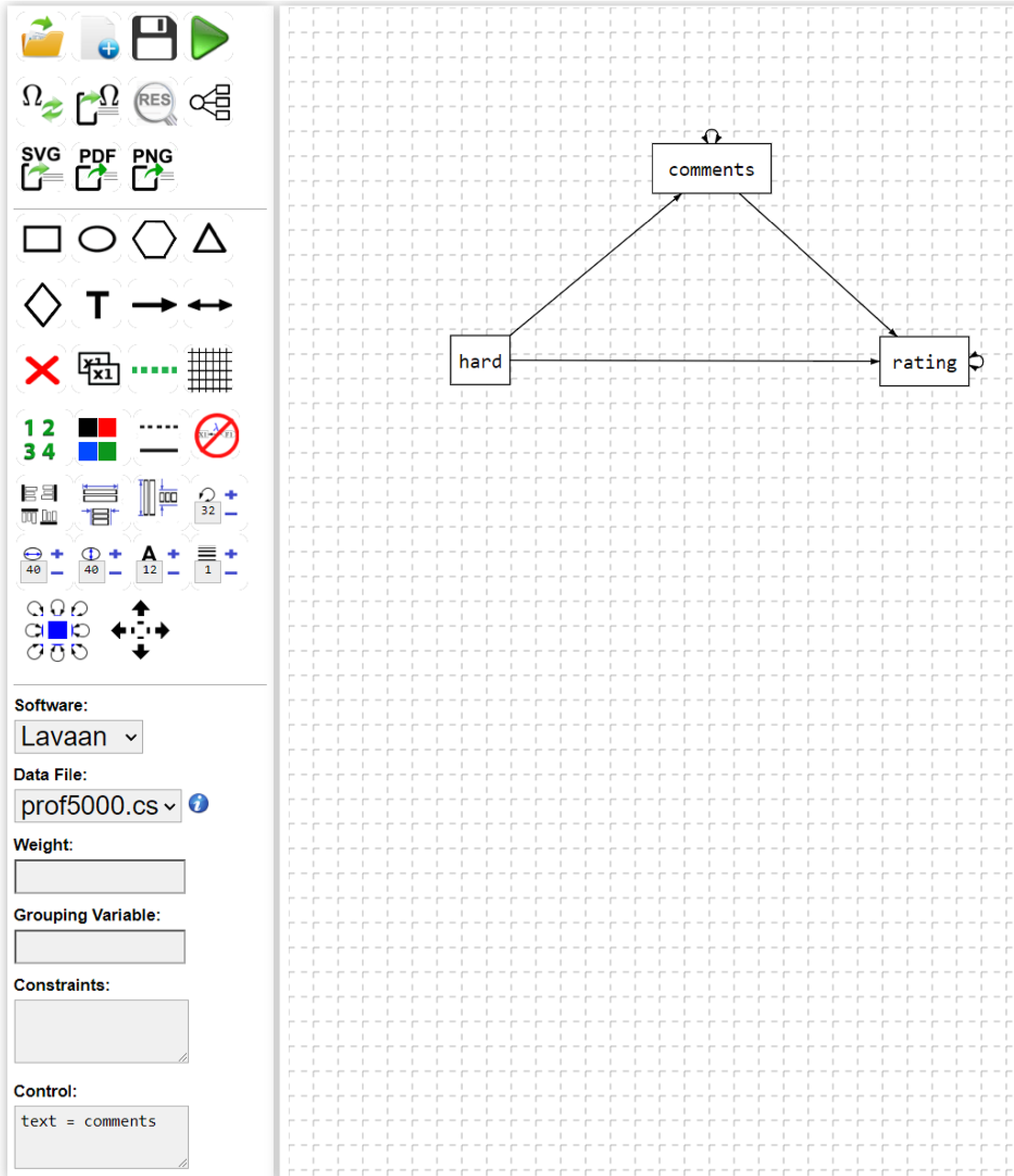
```
res <- sem.emb(sem_model = sem_model,
               data = prof.nest,
```

```
text_var = "comments",
pca_dim = 10,
encoder = "all-mpnet-base-v2")
```



Use of Web App

One can conduct the analysis by drawing a path diagram. To start, click the "Path Diagram" button. The interface below will appear:



A path diagram can be drawn through the buttons in the interface. In the example, we have a mediation model where the text is used as a mediator for the association of “hard” (how difficulty the class is) and “rating” (the numerical rating of the class).

Different from a regular SEM, we need to specify the variable "comments" as a text variable by setting "text = comments" in the "Control" field.

With that, one can click on the run button (the green arrow) to carry out the analysis. For example, for the current model, we have the output as below. It mainly has two parts - the data description and the model results.

Descriptive statistics (N=5000)

```
              Mean       sd     Min       Max    Skewness Kurtosis
id        1.4343e+04 8314.0453  9.0000 28521.000  5.7205e-03   1.7654
profid    4.8633e+02  299.9069  1.0000  1000.000  2.9661e-02   1.7294
rating    3.8618e+00    1.4581  1.0000     5.000 -9.5170e-01   2.4063
hard      2.8908e+00    1.3156  1.0000     5.000  5.7725e-02   1.8941
sentiment 2.0682e-01    0.2668 -1.4732     1.803 -6.3469e-04   4.6312
          Missing Rate
id                   0
profid               0
rating               0
hard                 0
sentiment            0
```

Model information

Observed variables: hard comments rating .

Text variables: comments .

The weight is: 0 .

The software to be used is: sem.text

```
lavaan 0.6-12 ended normally after 20 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                         9

  Number of observations                          5000
  Number of missing patterns                         1

Model Test User Model:

  Test statistic                                 0.000
  Degrees of freedom                                 0

Model Test Baseline Model:

  Test statistic                              4142.684
  Degrees of freedom                                 3
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    1.000
  Tucker-Lewis Index (TLI)                       1.000
```

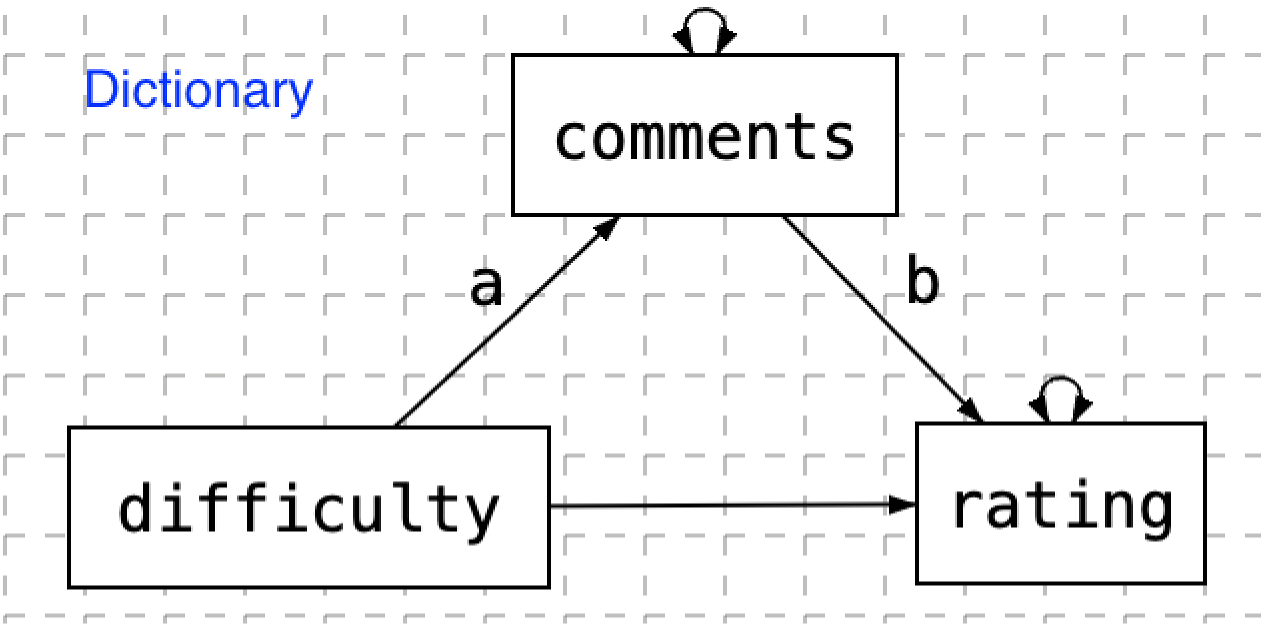
```
Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)             -15862.021
  Loglikelihood unrestricted model (H1)     -15862.021

  Akaike (AIC)                               31742.042
  Bayesian (BIC)                             31800.696
  Sample-size adjusted Bayesian (BIC)        31772.098

Root Mean Square Error of Approximation:

  RMSEA                                          0.000
  90 Percent confidence interval - lower         0.000
  90 Percent confidence interval - upper         0.000
  P-value RMSEA <= 0.05                             NA

Standardized Root Mean Square Residual:

  SRMR                                           0.000

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Observed
  Observed information based on                Hessian

Regressions:
                         Estimate  Std.Err  z-value  P(>|z|)
  comments.OverallSenti ~
    hard                   -0.075    0.003  -28.208    0.000
  rating ~
    cmmnts.OvrllSn          2.829    0.059   47.785    0.000
    hard                   -0.355    0.012  -29.605    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .cmmnts.OvrllSn    0.424    0.008   50.120    0.000
   .rating            4.304    0.043   99.150    0.000
    hard              2.891    0.019  155.389    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .cmmnts.OvrllSn    0.061    0.001   50.000    0.000
   .rating            1.076    0.022   50.000    0.000
    hard              1.730    0.035   50.000    0.000
```

Video tutorials text data analysis

Here we show how to conduct different types of analysis.

Mediation analysis with dictionary-based sentiment

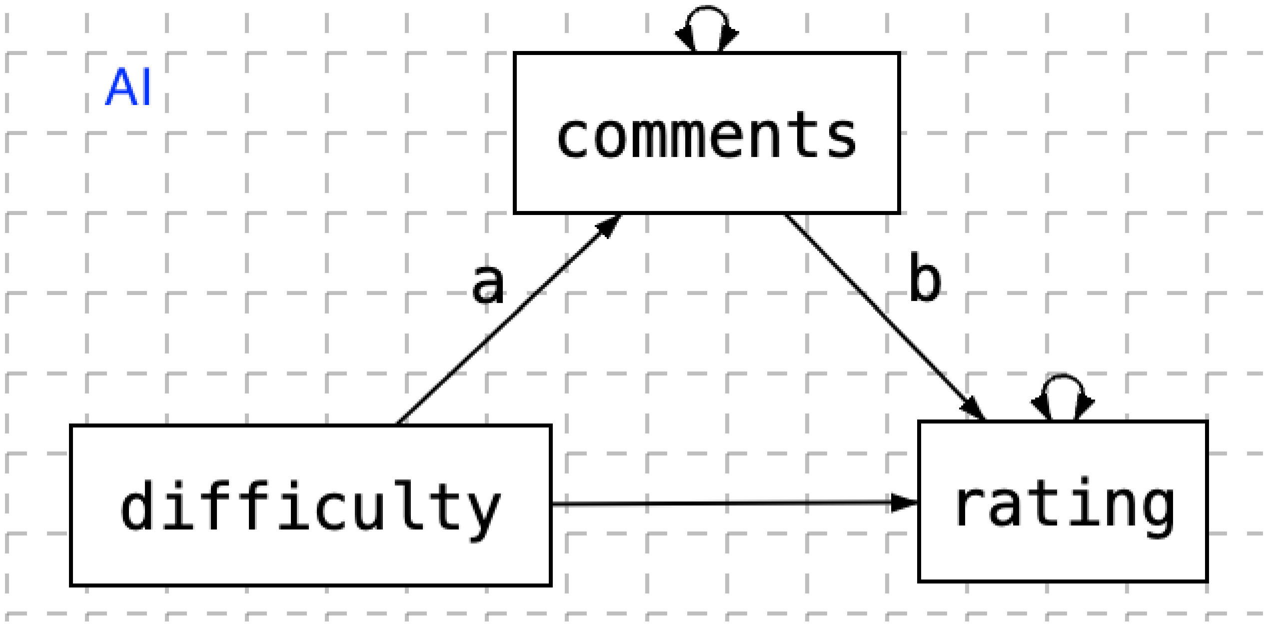
The model used here is



The video tutorial

Mediation analysis with AI-based sentiment

The model is



Factor analysis

In this example, we form a factor using two text variables - teaching comments and tags.