Supplementary Online Appendix

(Not for publication)

Deciphering Federal Reserve Communication via Text Analysis of Alternative FOMC Statements

Taeyoung Doh Dongho Song Shu-Kuei Yang

A Details about Text Analysis

A.1 Text Analysis using a Large Language Model

We briefly describe the procedure that we use to analyze text data using a large language model.[1] The raw text data is unstructured and can be represented as a string of characters, including letters, numbers, and symbols. We denote the total number of relevant characters as $|C|$. Consequently, each text is a string of characters with a variable length, as illustrated in the following example:

$$\text{Text}_t = [H, o, w, <>, a, r, e, <>, y, o, u, ?] \in |C|^{12}. \tag{A-1}$$

Because this data is unstructured, distinguishing different text data using a distance metric can be challenging. The embedding of text data by a large language model provides a representation of the original string data as a numeric vector, allowing us to define various text data features within a vector space. In the model, this embedding process involves two distinct steps: 1) tokenization, 2) neural network architectures.

A.1.1 Tokenization

Characters do not have linguistic or statistical meaning by themselves. A language model converts the sequence of characters into the sequence of tokens that are more interpretable linguistically or statistically. Language models transform the raw input text into the sequence of tokens by using tokenizers. One of popular tokenizers is PTB tokenizer, which obeys the

[1]Technical discussion in this section heavily draws on lectures notes on a large language model offered by the computer science department at Stanford University. For more details, see the above link and references therein.

English grammar and separates the contraction into two units. USE applies the PTB tokenizer to the raw input text. After the tokenization, the input data is transformed from the sequence of characters to the sequence of tokens, in which each token typically corresponds to a distinct vocabulary.

$$\phi : |C|^l \to |V|^L \, , \, \text{Text}_{v,t} = \phi(\text{Text}_t), \tag{A-2}$$

$$[H, o, w, <>, a, r, e, <>, y, o, u, ?] \to [How, <>, are, you, ?]. \tag{A-3}$$

The length of tokens in the sequence (L) might differ from the length of characters in the sequence (l) due to the pairing of characters during tokenization. Each token has its own embedding as a numeric vector. We can denote this token embedding as w. The result of tokenization is a sequence of token embeddings that represent the input text as $[w_1, \ldots, w_L]$. Each token is represented as a 64-dimensional numeric vector.

A.2 Neural Network Architecture

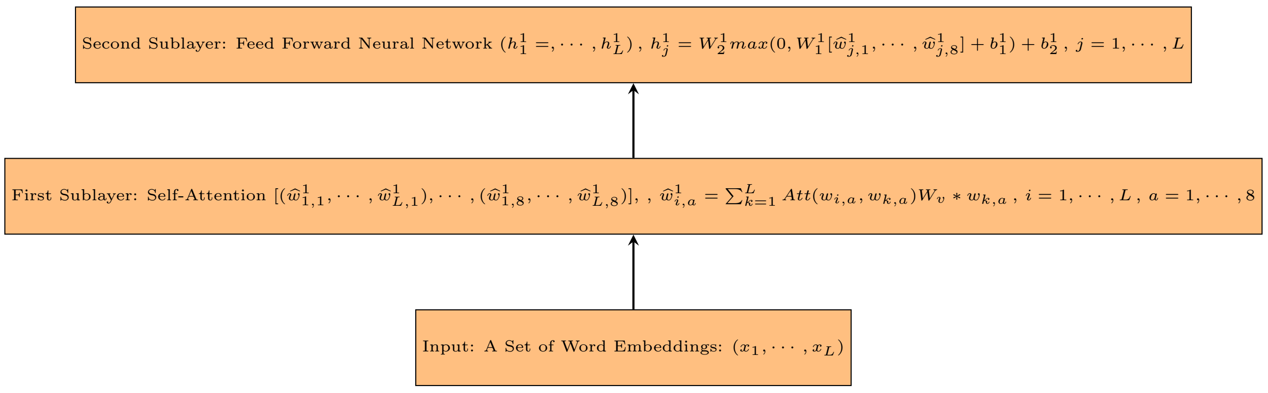
The USE has the transformer architecture consisting of six neutral network layers, each of which has two sublayers with eight self-attention heads. We describe tho original architecture and then explain how to fine tune it to obtain the paragraph level decomposition of similarity scoring across statements.

The first neural network in the USE is built by linking two sublayers as shown in Figure A-1 after taking a group of word embeddings that represents the source sentence as input. For the first layer, the same token embeddings are used for each attention head as input. In other words, $w^1_{i,a} = w_i \, , \forall a = 1, \cdots, 8$. The first layer generates the eight sequence of word embeddings $([\widetilde{w}^1_{1,1}, \cdots, \widetilde{w}^1_{L,1}], \cdots, [\widetilde{w}^1_{1,8}, \cdots, \widetilde{w}^1_{L,8}])$ as output and feeds this as input for the second layer. The actual USE architecture is slightly more complicated than presented below. It involves 1) positional embedding in which the order of any given word is also mapped into the embedding of that word,[2] 2) residual connection in which input bypasses attention and feed-forward neural network channels with a certain probability known as the dropout rate, 3) output from the layer is normalized to have mean zero and standard deviation of one.

The self attention channel can be best understood as looking up the dictionary value of all the words ($W_v * w_j \, , \forall j = 1, \cdots, L$) in the input text to match the query ($W_q * w_i$). The strength of match with the query word for each word in the look-up table is determined by the key ($W_k * w_j$). Here, attention weights for the a-th head are determined by how strong the key is with respect

[2]The position of each word embedding is used to generate since and cosine functions of different frequencies and these values are used the position embedding of the i-th word (P_i) and added to the input embedding w_i.

Figure A-1: First Neural Network Layer


Second Sublayer: Feed Forward Neural Network $(h_1^1 =, \cdots, h_L^1)$, $h_j^1 = W_2^1 max(0, W_1^1[\widehat{w}_{j,1}^1, \cdots, \widehat{w}_{j,8}^1] + b_1^1) + b_2^1$, $j = 1, \cdots, L$
First Sublayer: Self-Attention $[(\widehat{w}_{1,1}^1, \cdots, \widehat{w}_{L,1}^1), \cdots, (\widehat{w}_{1,8}^1, \cdots, \widehat{w}_{L,8}^1)]$, , $\widehat{w}_{i,a}^1 = \sum_{k=1}^L Att(w_{i,a}, w_{k,a})W_v * w_{k,a}$, $i = 1, \cdots, L$, $a = 1, \cdots, 8$
Input: A Set of Word Embeddings: $(x_1, \cdots, x_L)$

to the query word. In the attention sublayer, every word embedding is linearly transformed to have the key, the query, and the value representation through W_k, W_q, W_v.

$$\text{(A-4)} \qquad \widehat{w}_{i,a} = \sum_{k=1}^{L} Att(w_{i,a}, w_{k,a})W_v * w_{k,a} \ , \ Att(w_{i,a}, w_{k,a}) = \frac{e^{w'_{i,a}W'_k W_q w_{k,a}}}{\sum_{k=1}^{L} e^{w'_{i,a}W'_k W_q w_{k,al}}}.$$

Once word embeddings based on eight attention heads are obtained, USE concatenates these eight embeddings of each word into one big embedding and apply the feedforward neural network to this sequence of L embeddings. The final output from the second sublayer is the sequence of L embeddings.[3]

The second layer takes the output of the first layer as input and split the 512-dimensional vector representation (h_i^1) into eight 64-dimensional vector representations $([w_{i,1}^2, \cdots, w_{i,8}^2])$.

The entire USE algorithm works by vertically stacking six neural network layers which take the sentence embedding output in the previous layer as input and generate another sentence embedding as output. Figure A-2 describes the entire process.

To train parameters in the neural network architecture, we need to define the loss function that compares outcomes based on sentence embeddings from the USE with those based on human judgement. For example, if we define the relation between two texts as one of 3 classes (entail,contradict,neutral), we can apply the softmax classifier (f) to the difference between two embeddings. In this case, we can choose parameters in the neural network architecture to minimize the loss function that measures the distance between the machine-classified outcome $(f(U^i, U^j))$ and the one judged by humans $(f^{\text{human}}(\text{Text}_i, \text{Text}_j))$ where U^i is the 512-dimensional

[3]Each embedding representation has 512 dimension because we concatenate eight transformations of 64 dimensional original token embeddings.

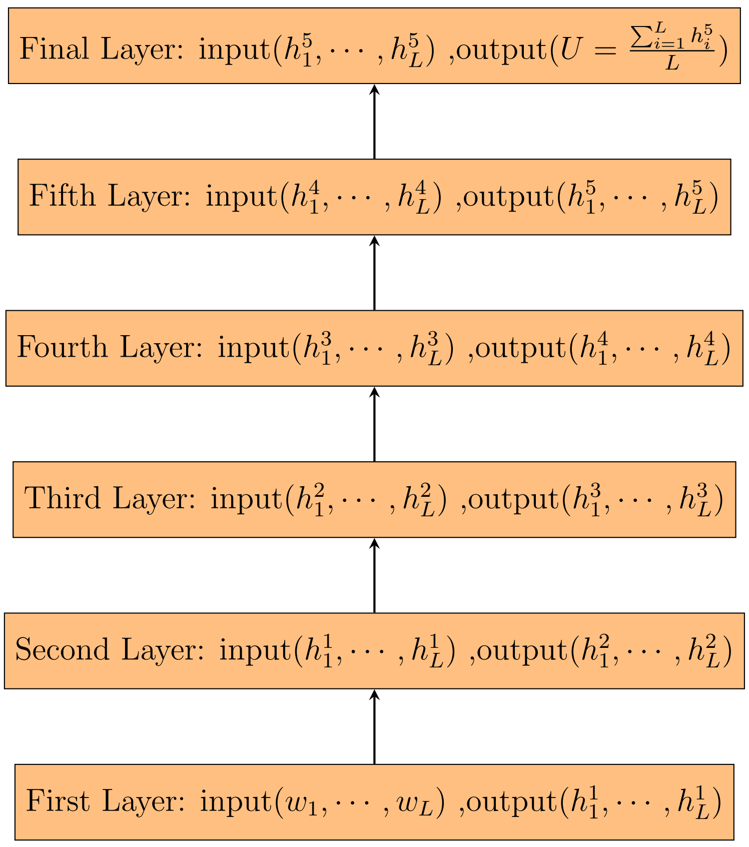

Final Layer: input$(h_1^5, \cdots, h_L^5)$,output$(U = \frac{\sum_{i=1}^L h_i^5}{L})$
Fifth Layer: input$(h_1^4, \cdots, h_L^4)$,output$(h_1^5, \cdots, h_L^5)$
Fourth Layer: input$(h_1^3, \cdots, h_L^3)$,output$(h_1^4, \cdots, h_L^4)$
Third Layer: input$(h_1^2, \cdots, h_L^2)$,output$(h_1^3, \cdots, h_L^3)$
Second Layer: input$(h_1^1, \cdots, h_L^1)$,output$(h_1^2, \cdots, h_L^2)$
First Layer: input$(w_1, \cdots, w_L)$,output$(h_1^1, \cdots, h_L^1)$

Figure A-2: Neural Network Architecture

USE representation of Text$_i$. In addition, two other natural language processing tasks are run to train the model.

- **Skip-thought task**: conditional on the center sentence, predict neighboring sentences (previous and next). The training dataset is from wikipedia articles.
- **Question-answer prediction**: predict the correct response for a given question among a list of correct answers and other randomly sampled answers. The training dataset is from web question-answer pages and discussion forums.
- **Natural language inference**: given a premise sentence and a hypothesis sentence, extract the relation between them. Let U_p and U_h be the sentence embeddings of the premise and the hypothesis, respectively. A fully-connect layer and and a 3-way softmax classifier are applied for the concatenated input of (U_p, U_h, $|U_p - U_h|$, and $U_p - U_h$). The three-way classifier predicts if the premise entails, contradicts, or is neutral to the hypothesis. The training dataset is the Stanford Natural Language Inference (SNLI) corpus.

A.3 Paragraph Level Decomposition of the USE Representation

In some cases, paragraph-by-paragraph comparison may provide more interpretable results. For instance, we may be interested in which paragraph drives the similarity score between different statements. For this, we obtain paragraph level USE representations and approximate the statement level USE representation by a weighted average of paragraph level USE representations.

Denote the USE representation of the released FOMC statement at time t by S_t^R. Similarly, S_t^i, $(i = A, B, C, D)$ denotes the USE representation of alternative statements. The USE representation of the j-th paragraph of the FOMC statement at time t is $P_{j,t}^i$. To calculate $P_{j,t}^i$, we run the USE algorithm for each paragraph j. The idea is to construct $\sum_k w_k P_{k,t}^i$ that can mimic S_t^i best in terms of minimizing the squared difference between two representations of the FOMC statement at time t.

- **Step 1: Paragraph Padding** Some statements are longer than others, meaning that the corpus of FOMC statements has an unequal length depending on the statement. An easy way to fix this is to pad a shorter statement with empty paragraph encodings. Suppose that n_{max} is the maximum number of paragraph of any given FOMC statement from the entire corpus of our dataset including both released statements and alternative statements. Then, we can extract the following array of the paragraph USE representation of the FOMC statement.

  $$P_t^R = [P_{1,t}^R, \cdots, P_{n_{max},t}^R]. \tag{A-5}$$

  If the number of paragraphs in the statement at time t $(n_{R,t})$ is smaller than n_{max}, we add $(n_{max} - n_{R,t})$ zero vectors of 512 dimensions. The purpose of this operation is to make the USE representation of any FOMC statement have the same number of the USE representations at the paragraph level.

- **Step 2: Approximate the Statement Level USE Representation by a Weighted Average of Paragraph Level USE Representations**

  The goal is to select weights $(w_j\,,\, j = (1, \cdots, n_{max})$ that can mimic this statement-level USE representation using paragraph-level USE representations. We consider the following squared loss:

  $$\sum_{i \in R,A,B,C,D} \sum_t (S_t^i - \sum_j w_j P_{j,t}^i)^T (S_t^i - \sum_j w_j P_{j,t}^i). \tag{A-6}$$

We can put the non-negativity and unit-sum constraints on w_j such that $w_j >= 0$, $\sum_j w_j = 1$. Once we find the solution for weights, we can mimic P_t^i by $\sum_j w_j P_{j,t}^i$. But the numerical optimization routine might be non-convex when you put the constraints directly. So we may consider the following transformation of w_j to make the problem an unconstrained minimization problem:

$$w_j = \frac{e^{\alpha_j}}{\sum_{k=1}^{n_{max}} e^{\alpha_k}}, \tag{A-7}$$

where α_j is an unconstrained parameter. Notice that w_j still satisfies the constraints but we are minimizing the loss function with respect to $(\alpha_1, \cdots, \alpha_{n_{max}})$.

- **Step 3: Decomposing the Similarity Scoring**

  For the unit-vector, the cosine similarity is simply the inner product. So we can renormalize the USE representation to have a unit length. In that case, we have the following nice decomposition of the similarity scoring between texts.

$$Sim(P_t^i, P_t^j) \propto Sim(\sum_{k=1}^{n_{max}} w_k P_{k,t}^i, \sum_{k=1}^{n_{max}} w_k P_{k,t}^j) = \sum_k \sum_{k'} w_k w_{k'} Sim(P_{k,t}^i, P_{k',t}^j). \tag{A-8}$$

A.4 Details of Fine-tuning

As explained in the text, we add an additional layer to the USE representation of the text to train the final embedding output to recognize numeric properties better. We consider a fully connected feed-forward network with a rectified linear unit as an activation function. For the original USE representation of a FOMC statement $U_{FOMC} = [U_1, \cdots, U_{512}]$, our additional layer performs the following transformation:

$$f(U_{FOMC}) = [\max(W_1' U_{FOMC} + b_1, 0), \cdots, \max(W_{512}' U_{FOMC} + b_{512}, 0)]. \tag{A-9}$$

Let's stack parameters governing this transformation by $\vartheta = [W_1, \cdots, W_{512}, b]$ where $b = [b_1, \cdots, b_{512}]$. As described in the text, we generate two separate training datasets to optimize ϑ in order to minimize loss functions set out in equation (2) and (3).

B Prediction Regression

Consider the following linear regression model:

$$y_t = z'_{t-\Delta}\delta + e_t, \quad e_t \sim \mathcal{N}(0, \sigma^2), \tag{A-10}$$

where $y_t \in \mathbb{R}$ and $z_{t-\Delta} \in \mathbb{R}^k$ is the full predictor vector observed at time $t-\Delta$. In the presence of missing observations in $z_{t-\Delta}$, we define a row selection matrix $M_{t-\Delta} \in \mathbb{R}^{k_{t-\Delta}\times k}$, with $k_{t-\Delta} < k$, that extracts only the observed components of $z_{t-\Delta}$. In this case, we can re-express (A-10) as:

$$y_t = x'_{t-\Delta}\delta + e_t, \quad e_t \sim \mathcal{N}(0, \sigma^2), \tag{A-11}$$

where $x'_{t-\Delta} \equiv z'_{t-\Delta}M'_{t-\Delta}M_{t-\Delta}$, and $M'_{t-\Delta}M_{t-\Delta} \in \mathbb{R}^{k\times k}$ is a diagonal matrix that zeroes out unobserved elements of $z_{t-\Delta}$.

B.1 Posterior updating rules

This formulation of (A-11) ensures that only the observed entries contribute to posterior learning.

Prior distributions. We assume the following prior distributions:

$$\delta \mid \sigma^2 \sim N(\delta_{t-1|t-1}, \sigma^2 V_{t-1|t-1}), \quad \sigma^2 \sim \mathrm{IG}\left(\frac{\nu_{t-1}}{2}, \frac{\nu_{t-1}s^2_{t-1}}{2}\right). \tag{A-12}$$

Given a new observation $(y_t, x_{t-\Delta})$, the likelihood function is:

$$p(y_t \mid \delta, \sigma^2, x_{t-\Delta}) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{(y_t - x'_{t-\Delta}\delta)^2}{2\sigma^2}\right).$$

Posterior distributions. The posterior distribution of σ^2 follows an Inverse-Gamma

$$\sigma^2 \mid y_t, x_{t-\Delta} \sim \mathrm{IG}\left(\frac{\nu_t}{2}, \frac{\nu_t s^2_t}{2}\right), \tag{A-13}$$

with the update as follows:

$$\nu_t = \nu_{t-1} + 1, \quad \nu_t s^2_t = \nu_{t-1}s^2_{t-1} + \frac{(y_t - x'_{t-\Delta}\delta_{t-1|t-1})^2}{1 + x'_{t-\Delta}V_{t-1|t-1}x_{t-\Delta}}.$$

Conditionally on σ^2, the posterior distribution of δ remains normal:

$$\delta \mid y_t, x_{t-\Delta}, \sigma^2 \sim N(\delta_{t|t}, \sigma^2 V_{t|t}), \tag{A-14}$$

where the posterior mean and variance update as follows:

$$\delta_{t|t} = V_{t|t}\left(V_{t-1|t-1}^{-1}\delta_{t-1|t-1} + x_{t-\Delta}y_t\right), \quad V_{t|t} = \left(V_{t-1|t-1}^{-1} + x_{t-\Delta}x_{t-\Delta}'\right)^{-1}.$$

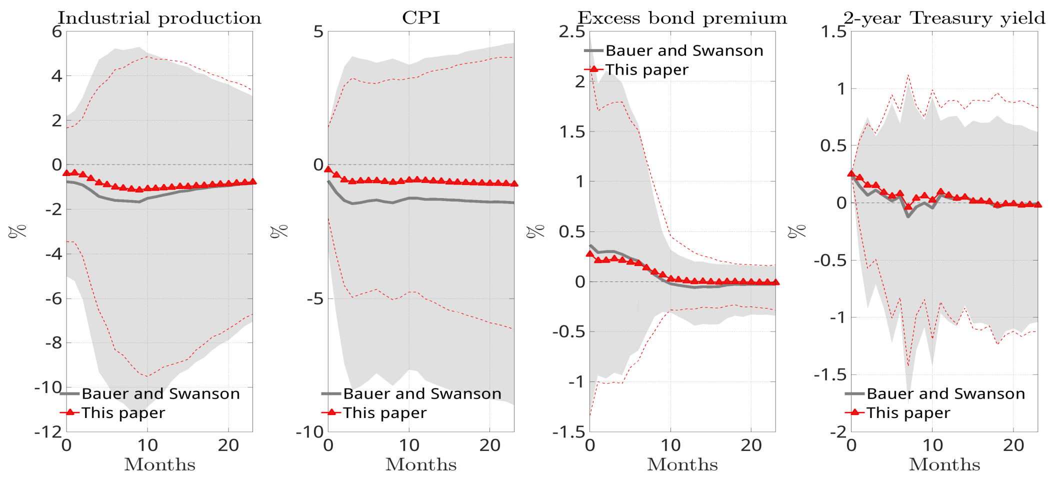
C Residual-based Moving Block Bootstrap (MBB) Confidence Intervals for SVAR IRFs

Confidence intervals for impulse response functions in Figure 3 are computed by the wild bootstrap method used in Bauer and Swanson (2023a). Jentsch and Lunsford (2022) show that the method is not asymptotically valid and can result in misleading confidence intervals and coverage rates for time series data with serial dependence. As an alternative, they propose a residual-based moving block bootstrap (MBB) method that can address these issues. We recalculate 90% and 68% confidence intervals for SVAR impulse-responses in the main text using the residual-based moving block method in Figure A-3.

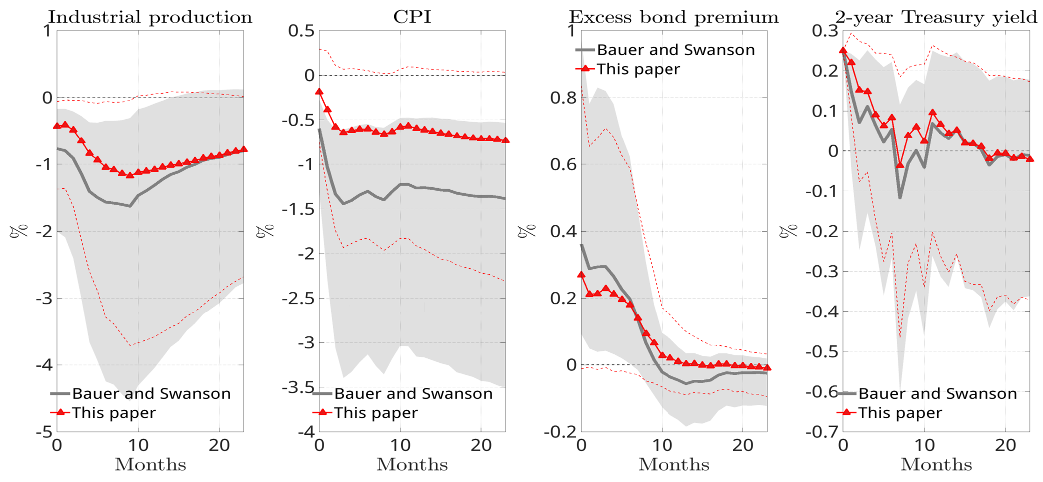
Compared to 90% confidence intervals in Figure 3 based on the wild bootstrapping method, we observe wider bands in general. Both monetary policy shocks identified by the method in Bauer and Swanson (2023a) and our method lose statistical significance in 90% confidence bands when we use MBB. Nonetheless, a contractionary monetary policy shock reduces industrial production and CPI while increasing the excess bond premium in a statistically significant way if we focus on the 68% confidence interval in Figure A-3. Point estimates based on the median IRFs are robust to the use of different bootstrapping methods.

Figure A-3: SVAR IRFs with MBB confidence intervals

Panel A. 90% Confidence Interval



Panel B. 68% Confidence Interval



Notes: Following Bauer and Swanson (2023b), we estimate a SVAR with a 12-month lag using the following variables: log industrial production, log consumer price index, excess bond premium Gilchrist and Zakrajšek (2012), and the two-year Treasury bond yield. To instrument our analysis, we employ the orthogonalized monetary policy surprise measure derived from the residuals of a regression on six predictors explained in Table 4 observed before the FOMC announcement. While we estimate the VAR coefficients using a longer sample from 1973:M1 to 2019:M12, the orthogonalized high-frequency monetary policy surprises are available only from 2004:M3 to 2016:M12. To facilitate comparison, we present the impulse responses obtained by utilizing the orthogonalized high-frequency monetary policy surprise instrument employed in Bauer and Swanson (2023b). Confidence intervals are calculated based on the residual-based moving block bootstrapping method in Jentsch and Lunsford (2022).