Assessing policy sentiment with Natural Language Processing methods

by
Kelston Chen

An essay submitted to the Department of Economics
in partial fulfillment of the requirements for
the degree of Master of Arts

Queen's University
Kingston, Ontario, Canada
August 2023

Abstract

The growing prevalence of AI, in *Natural Language Processing* (NLP), and the heightened importance of managing expectations, with monetary policy, have opened new doors in the world of economic forecasting. By adopting a *hybrid approach* that combines *supervised* and *unsupervised* learning models, I successfully extract sentiment from the Bank of Canada's *Monetary Policy Reports (MPRs)* with remarkable accuracy, achieving an accuracy rate of 79 percent. Moreover, the extracted sentiment is further normalized, creating a distinctive feature for economic forecasting. The forecast results demonstrate that sentiment from the Bank of Canada provides novel information and emerges as the strongest predictor of monetary policy decisions (e.g., *Canadian Overnight Repo Rate Average*) and wider reaching interest rates (e.g., mortgage rates). The methods employed in this study show that, not only can text be used in a forecasting application, but that this process can be used to construct an efficient *data pipeline*, allowing for significantly faster application in future forecasts.

Acknowledgements

I would like to thank Professor Huw Lloyd-Ellis for his insight, support, and willingness to even accept me as a supervisee for this paper. I also give thanks to the QED program for its interesting set of elective courses (namely ECON 882, also known as "The Machine Learning Course", which fuelled my continued interest in machine learning) and for teaching me that I know less about economics than I initially thought possible. Outside of academics, I am grateful for the opportunities I've been given from attending the QED program and look forward to starting my career in September. I extend my gratitude to my family, friends, and (especially) my peers in the MA program. I thank them for creating such an enjoyable environment and a comradery which I could not have found anywhere else. Lastly, I would like to thank my partner and best friend Sarah Rabi, for enduring my frustrated objections toward the QED program during my enrollment. I now find that the end to these eventful 12 months is bittersweet. And to end on a famous, and made-up, quote falsely attributed to Dr. Seuss, "Don't cry because it's over; smile because it happened." - *Ludwig Jacobowski, 1899 (paraphrased)*.

Table of Contents

1. Introduction

The explosive takeover of generative AI, like ChatGPT, has abounded in the media and has significant implications for virtually all industries. Following this excitement, there have been massive strides in the capabilities of machine learning and deep learning models to understand human language, a field known as *Natural Language Processing* (NLP). Although the excitement around NLP may be more recent, its usage in the field of economics is far from new. For instance, early papers like Coase (1960), analyzed how the law can resolve externality issues, while Friedman and Schwartz (1963) created a measure of "policy surprises" with historical text documents (Ash & Hansen, 2023). More recently, the focus of NLP in economics has largely shifted towards deciphering meticulously written speeches and reports from monetary authorities around the world. The interest in using machines to analyze human text has increased at the same time with the importance and weight that central banks place on managing expectations with monetary policy (Blinder et al., 2008). Therefore, a natural case for using NLP in economics is to harness the power of *sentiment analysis* in *forecasting*. This observation motivates with this paper's objectives which are (i) to assess whether statistical models can extract sentiment from the Bank of Canada's, quarterly, *Monetary Policy Reports* (MPRs) and (ii) to examine if the extracted sentiment will have predictive power and statistical significance on changes in interest rates.[1] Overall I show how, NLP can provide a fast route around the laborious task of digesting economic text, with the methods in this paper able to be utilized in the creation of an efficient system (i.e., a data pipeline) for implementing text in forecasting tasks.

1.1 Literature review

Disentangling effects of monetary policy have been a challenge for of economists since the inception of modern central banking. In the influential Romer and Romer (2004) article, information from the Federal Reserve's "Greenbook" was used to measure the differences between observed and "intended" changes in the Federal Funds Rate, where the difference was labelled as monetary policy shocks (i.e., regression residuals). A subsequent paper from Aruoba and Dreschel (2023) expanded on this idea by not only using the numerical forecasting information from Fed documents but also the text itself. To do this, Aruoba and Drechsel applied standard NLP techniques to frequently used "concepts" – or words – and performed an *aspect-based approach* to assigning positive or negative

[1] Interest rates used will be the CORRA (Canadian Overnight Repo Rate Average), the prime rate and 1, 3, and 5-year mortgage rates.

sentiment to each concept with the use of a popular financial lexicon from Loughran and McDonald (2011).[2] The advantage of an aspect-based approach is its *unsupervised* nature,[3] which allows the authors to run real time analyses, something that Romer and Romer's (2004) paper could not have done.

In addition to dictionary-based approaches, many researchers have applied unconventional techniques which rely on *machine learning* have also been used. Kalamara et al., (2021) harness the abundance of newspaper articles in the UK to predict various macroeconomic variables. They find that employing text features improves forecast results both unconditionally and conditionally on other predictors. In addition to using the text as a predictor, *feature engineering* allows for even greater model performance through the addition of simple word counts as a regressor. The results of their machine learning models also suggest the influence of "animal spirits" on the macroeconomy with news articles remaining an effective predictor for up to 9 months (Keynes, 1936; Shiller, 2017; Kalamara et al., 2021). Moreover, the authors find that the use of *supervised* machine learning tends to perform better than dictionary-based methods.[4] This result also falls in line with work by Frankel et al., (2022), which find that machine learning methods outperform dictionary-based methods and are easier to implement when measuring sentiment from financial texts – 10-K filings. More specifically, both works find that the accuracy of machine learning models improve over time and during times of greater uncertainty. This provides evidence that supervised machine learning models are more flexible and adaptable to changes in language than dictionary methods.

In the Canadian context, Binette and Tchebotarev (2019) analyzed MPRs from 1995 to 2018. Unlike the previous papers, they do not use economic/financial text as a predictor for a macro indicator. Instead, Binette and Tchebotarev utilize more advanced *deep learning* techniques, something far less common in economic papers.[5] More specifically, the authors take advantage of a growing technique in NLP called *Transfer Learning*.[6] This allows a highly complex model to be *fine-*

[2] Aspect-based methods match words against a pre-defined dictionary or lexicon of words. A sentiment score can then be calculated based on the aspect – or concept or word – of interest. This method type falls under dictionary based methods. For more see, Wankhade et al. (2022).

[3] Unsupervised Learning is a method that does not require the use of a "ground truth" dependent variable to measure a model's loss. Thus, unsupervised models do not require training.

[4] Supervised Learning is the opposite of unsupervised learning. This approach is most similar to regression analysis, where the relationship between the dependent variable is compared to the regressors.

[5] Deep learning is a subset of Machine learning. In this context, machine learning usually refers to statistical learning methods – like logistic regression and Random forests – and deep learning refers to neural networks, a term that most refer to when thinking about AI and generative models like GPT.

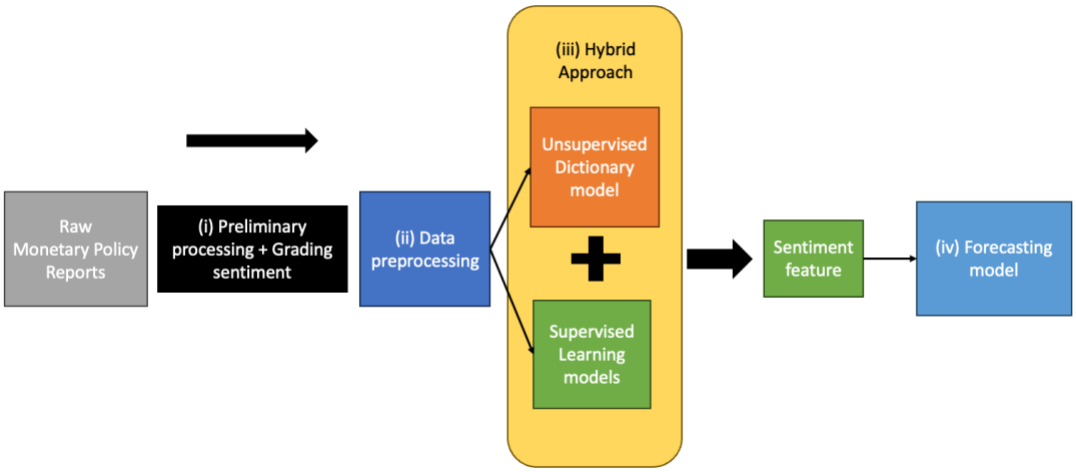
[6] Transfer Learning applies a pre-trained model – a model which has been trained a large amount of "general" text data (e.g., the entirety of Wikipedia) – on to a specific task. This method has been shown to be extremely effective and is a leading method in sentiment analysis (Howard & Ruder, 2018).

tuned to their specific task (sentiment analysis on economic text), significantly improving performance. With no forecasting element, this work focuses on breaking down the characteristics – readability, similarity/lexical innovation, and sentiment – of the MPRs over time.

Adding to the Canadian context, this paper applies NLP methods to the Bank of Canada's quarterly MPRs to assess sentiment at the *sentence* level. Building upon the sentiment analysis done by Binette and Tchebotarev, this paper explores the use of several different supervised learning methods and expands upon their work by, (i) including more recent reports from 2019Q1 to 2023Q2 and (ii) applying predicted sentiment from NLP models in a forecasting exercise. By implementing predictive text models, the ease and speed with which MPRs – or other economic text – can be utilized in subsequent economic applications, such as forecasting, improves significantly.

2. Methods overview

Figure 1: Outline of study (data to model pipeline)



Notes: The following provides a general outline of the steps taken and the order of the sections within the paper. In the above, (i) refers to section 3, (ii) is section 4, (iii) refers to sections 5-7, and (iv) is section 9.

The remaining sections of this paper is broken down into four main steps, (i) the preliminary tabulation of MPRs and sentiment grading;[7] (ii) cleaning and preprocessing of the text data (i.e., transforming text into numbers); (iii) sentiment analysis with supervised and unsupervised learning models, known as a *hybrid approach* (Wankhade et al., 2022); and (iv) using the predicted sentiment as a predictor for changes in interest rates.

[7] Sentiment will be classified into three classes: negative, neutral, and positive, and will initially be done by human reference.

3. Data collection and preliminary processing

This section outlines the steps taken to convert the MPRs into a tabular format and covers the basic methodology for sentiment grading.[8] As previously stated, this study uses MPRs from 2000Q1 to 2023Q2. Note that while MPRs from as early as 1995 are available for download, the switch in reporting frequency – from semi-annual releases to quarterly releases – required dropping the earlier MPRs, for consistency.

Before sentiment can be analyzed, the text data must go through many processing steps. The first of which requires converting the MPRs from a PDF file into a text file.[9] Since the conversion leaves fragments of non-translatable objects, large parts of the introductory pages, titles, chart elements, and any non-standard text objects, were hand removed. An automated cleaning program, provided by Binette and Tchebotarev's *GitHub* repository, was also used for cleaning.[10] Once the MPR text files were clean, each MPR was broken down into sentences using the *spaCy sentence parser* and exported in a tabular format (CSV file) where each row provided one sentence of text.[11] Focusing on a sentence level analysis, rather than a full MPR level analysis, allowed for a larger sample size and, thus, more training data. Analyzing at a per MPR level would have only given us 90 observations (2000Q1 to 2023Q2), while sentence level parsing provides us with over 17,000.

3.1 Grading sentiment

Once all MPRs are exported as a CSV file, human intuition is employed to assign sentiment to each sentence. Given three sentiment options – negative, neutral, and positive – each sentence is classified according to how the sentence made the reader feel about the outlook of the Canadian economy. For example, "Export volumes edged down, however, mainly owing to a marked slowing in the growth of US aggregate demand and to supply problems in the energy and automotive sectors.", would be a negative sentence. Bearish attitudes and uncertain outlooks were also common characteristics of negative sentences. Neutral sentences tend to be objective sentences or sentences which contain both positive and negative statements. For example, "The Bank continues to closely monitor a wide range of indicators of pressures on capacity and inflation.", is an objective and neutral statement. Lastly, a positive sentence will lead the reader to have a positive outlook on the economy; dovish statements would also fall under this category. The following, "Economic

[8] Sentiment was graded into three classes: negative, neutral, and positive, and will initially be done by human reference.

[9] PDFtoText was used to convert the publicly available Monetary Policy Reports: https://pdftotext.com/

[10] Binette and Tchebotarev's GitHub Repository: https://github.com/bankofcanada/MPR-Text-Analytics-2019

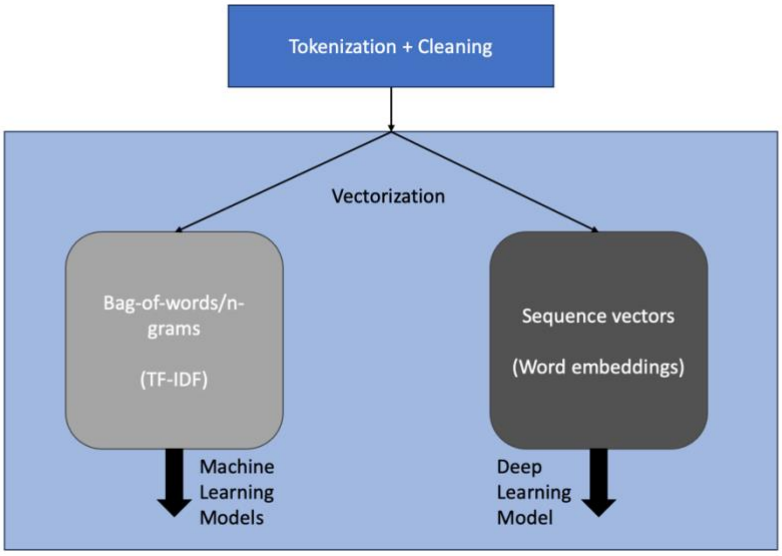
[11] spaCy is an open-source Python package which provides access to a system of pre-trained NLP models.

fundamentals continue to support the Canadian dollar, since growth remains robust, and the current account has moved into surplus.", is an example of a positive sentence.

It should be noted, however, that while the grading process adheres to general guidelines, it remains a challenging task and is not immune to errors. For example, sentences about rising inflation could register to readers as negative if the economy begins "overheating" but could be seen as a positive if the economy has been running below the target rate. Grading sentiment also became more challenging if sentiment was dependent on perspective, an issue noted by Consoli et al. (2022). For example, a sentence describing higher commodity prices may present a positive outlook for Canadian producers but pose a drag on consumer purchasing power. Thus, intuition was relied upon to understand the economic context of which the sentence belonged to when assigning sentiment.

4. Data preprocessing

Figure 2: Data preprocessing flow chart



The following section will cover typical text processes steps and methods to convert text (sentences) into a numerical representation – *tokenization*, *vectorization*, and *word embeddings*.[12]

[12] The main explanations of how these methods work are explained in the following subsections, more technical portions are presented in the appendix (A & B).

At the start of any NLP task, the text data must be cleaned and formatted so that a machine is able to interpret the data. The main idea behind any preprocessing step in NLP is to clean the data of noise that could impair learning and generate a numerical space that will represent the text data. The following subsections will follow the processing steps shown in figure 2.

4.1 Tokenization

The first step in turning text into numbers is to tokenize the text. Tokenizing is simply the process of dissecting a body of text – known as a *document* – into individual words – *tokens*.[13] For example, we can have the following document: *"The cat chased the mouse away"* which, after tokenizing, would become a list of tokens, *['The', 'cat', 'chased', 'the', 'mouse', 'away']*. When tokenizing there is also a choice of how many words to have in a single token, referred to as *n-grams*. This example made use of *unigrams* (n = 1), however, *bi-grams* (n = 2) or *trigrams* (n = 3) may be preferred if the surrounding words are relevant to the meaning or sentiment of the document. Using the same example, a bigram would create, *['The cat', 'cat chased', 'chased the', 'the mouse', 'mouse away']*.

4.2 Text cleaning

Once the *corpus* – the total aggregation of documents – has been tokenized, further cleaning needs to be done. Although some cleaning was completed in the preliminary steps, many less obvious features of the text still needed to be cleaned. This included *capitalization*, *punctuation*, *numbers*, *stop words*, and *negation*.

Forcing all tokens to be lower case acts to *normalize* the text, reducing the dimensionality of the text data and boosting model performance (Vernikou et al., 2022).[14] Punctuation is also excluded from the text since it often doesn't contribute significant additional information compared to the introduced noise (Kowsari et al., 2019). Additionally, given the uniform and non-provocative writing style of MPRs, punctuation is even less likely to provide relevant information. For similar reasons, numbers were also removed, reducing noise and improving model performance (Jianqiang & Xiaolin, 2017; HaCohen-Kerner et al., 2020).

Stop words and negation was dealt with using a pre-defined list of stop words – from the *Python* package *NLTK (Natural Language Toolkit)* – and a list of negative words – from Loughran and

[13] In the NLP field, a "document" refers to any unit (e.g., aspect, sentence, paragraph, or document/file) of the "corpus" – the combined body of documents.

[14] Text normalization works to reduce the number of times the same base word appears, reducing the dimensionality of the data (e.g., "the" and "The" would not be understood as the same word by a computer).

McDonald (2011) and Böök et al. (2019). In NLP, stop words refer to a set of commonly used words which do not evoke much meaning and are usually removed from the corpus (Kaur & Buttar, 2018). Some examples of stop words could be, "the", "a", "as", and "was", although no single universal list, or standard length, of stop words has been defined (Manning et al., 2008). To address negation, a list of words was compiled that may switch a positive sentence to a negative one. As an example, the word "didn't" would act as a negative word in the following sentences, *"I did eat the sandwich"*, versus, *"I didn't eat the sandwich"*. Recognizing their importance to text sentiment, negative words were allowed to stay in the corpus as negation is likely to show up in economic and financial text (Loughran & McDonald, 2011). Once the corpus is cleaned of stop words and other unnecessary text features, the remaining tokens became part of the *vocabulary*.[15]

4.3 Vectorization – bag-of-words models

For the models to understand human text, it must be translated into their mother tongue –numbers. This step is commonly referred to as vectorization since the token lists – created above – will be represented as vectors. The initial step is to assign an index number to each token. For example, the sentence, *"The cat chased the mouse all around the house until it got caught"*, would be assigned the following index numbers:

Index numbers for words in "The cat chased the mouse all around the house until it got caught" – the vocabulary

9	2	4	8	0	1	6	10	7	5	3
the	cat	chased	mouse	all	around	house	until	it	got	caught

While the mapping of words to index numbers is the vocabulary for the above example, in the literature there are various ways of vectorizing text data. The most elementary way to vectorize is through *one-hot encoding*.[16] This method assigns a 1 if the word is present and 0 otherwise. As an example, given the vocabulary we created above, if the sentence, *"The cat chased the mouse"*, was one-hot encoded, the representative vector would look like the following:

"The cat chased the mouse" one-hot encoded

[0 0 1 0 1 0 0 0 1 1 0]

The appearance of a 1 can be understood by mapping the position in the encoded vector with the index number assigned to each token. Therefore, the first position in the vector is 0, since 0 maps to the token "all", which does not appear in *"The cat chased the mouse"*. *Count encoding* assigns

[15] In NLP, vocabulary refers to the full list of words/tokens, after cleaning, that exist in the text data being used.

[16] One-hot encoding is synonymous to using "dummies" in econometrics.

a 0 or an integer greater than or equal to 1 (for the number of times that token appears in the text). By counting the number of appearances, we can give greater weight to more frequent words.

These methods are often referred to as *bag-of-words* (BoW) models because the order of the words does not matter (Bird et al., 2009).

4.3.1 Term frequency-inverse document frequency (TF-IDF)

TF-IDF has become a standard BoW model. This method extends upon count encoding by recording both the *term frequency* (TF) – same as count encoding – and the *inverse document frequency* (IDF). In contrast to TF, IDF considers rare words more informative – assigning more weight to infrequent words and less weight to frequent words. This makes words like "the" carry little weight in a sentence. The resulting TF-IDF vector is the product of TF and IDF (see appendix A). Using the same example as before, TF-IDF vectorization produces the following output vector:

"The cat chased the mouse" TF-IDF encoded

[0 0 0.378 0 0.378 0 0 0 0.378 0.756 0]

It is important to note that although the output vectors from these examples are small, with real data the outputs from vectorization lead to large (high dimension) and sparse matrices (James et al., 2021). Noting this, the output matrix from TF-IDF vectorization on the training data of MPR text led to a 12,121 × 4,760 sparse matrix. The improved information extraction offered with TF-IDF vectorization led to its usage in all machine learning models in this study.

4.4 Sequence vectors

While the above methods are categorized as BoW models, due to their lack of information on word order, deep learning methods can utilize the order of words to understand context. In order to capture word order, sequence vectors, and word embeddings, are used. However, an issue with sequence vectors is that they can be large and sparse – known as a dimensionality issue (see appendix B). In our running example, the sentence, *"The cat chased the mouse"*, would return four sparse vectors rather than one, like before. [17]

4.4.1 Word embeddings

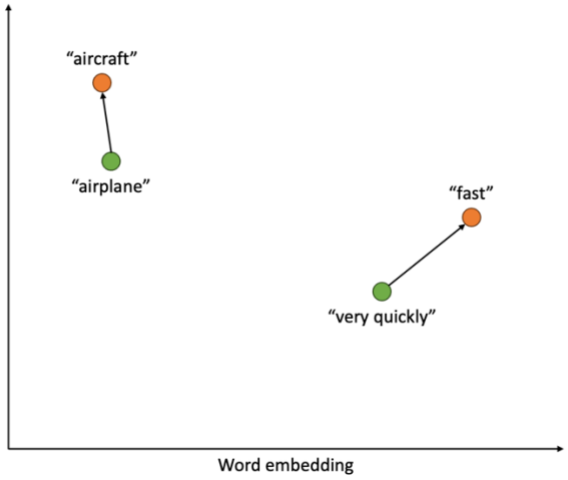
Word embeddings solve the dimensionality issue, because it can capture similarities between pairs of words and implicitly understand semantics (Kowsari et al., 2019). For example, although

[17] A common negative side effect of large sparse matrices is slower training times and lower model performance.

sentences like, “An airplane flew by very quickly” and, “A fast aircraft flew by” are different, they convey the same idea. Word embeddings are able to create denser and smaller matrices because it can capture the similarities between the words, like “airplane” and “aircraft” (see appendix B; figure 3), whereas BoW would return two completely different vectors.

There are two commonly used word embedding models, *Word2Vec* (Mikolov et al., 2013) and *GloVe* (Pennington et al., 2014), both of which will be used in this paper. Namely, a pre-trained Word2Vec model will be used to calculate similarity, using the *Word Mover’s Distance (WMD)* metric, and a pre-trained GloVe embedding layer will be used in our deep learning model.

Figure 3: Word embeddings

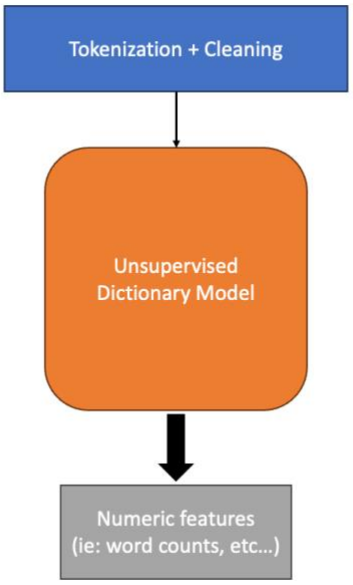


Notes: This is an example of a word embedding from the two sentences: “An airplane flew by very quickly” and “A fast aircraft flew by”, in a 2-D space. In practice, the dimensions of the word embedding space is much larger (i.e., n >2). Re-created and adapted from Chollet (2017).

5. Unsupervised Model

The following sections will highlight the models used in the hybrid approach. As stated, the hybrid approach will use an unsupervised and supervised learning model to learn and detect sentiment. This involves the use of a dictionary-based approach along with a machine learning, or deep learning, model to predict sentiment. Dictionary methods are natural starting point since they do not require any training.

Figure 4: Dictionary model flowchart



5.1 Dictionary method

Although dictionary methods are not accurate predictors alone, it can allow us to quickly extract information and gather data characteristics from the MPRs. The dictionary or lexicon used in our model will be taken from Loughran and McDonald's (2011) updated financial lexicon. This lexicon was built using 10Ks from 1994 to 2008, categorizing over 86,000 unique words into one of seven categories – negative, positive, uncertain, litigious, strong modal, weak modal, and constraining. Loughran and McDonald's lexicon was preferred over widely used lexicons like the *Harvard psychology dictionary* given the semantic difference of words in financial contexts.

The dictionary model used follows a simple rule-based algorithm. Each word in a sentence, was checked against the categories from the financial lexicon.[18] If a match was found, a count would be added to the appropriate category. In the special case of negation, the algorithm would check for negative words – from a list generated by Böök et al. (2019) – within three words of a positive word. If a negative word existed within the range, then the count would be given to the negative category rather than the positive. For example, a phrase like, "strong growth" would be counted as two positive words, but "strong decline" would be counted as two negatives. This accounts for simple negation but ignores the case of a double negative, an unlikely case for financial and economic texts

[18] All categories but "litigious" were used.

(Loughran & McDonald, 2011). As a result, the dictionary method produces a list of counts for each category (table 1).

5.1.1 Data characteristics

Table 1 shows the summary statistics calculated from the dictionary algorithm. On average we find that negative words were more prominent, per sentence, than positive words. Words that convey uncertainty were another common feature in the monetary text. Interestingly, our results are similar to that found by Loughran and McDonald from their analysis of 10Ks. The output of the model also allows us to look at some of the most common words from each category (table 2). We can also use a word cloud to represent frequently used words or phrases from the past 22 years of monetary reports.[19] As expected with policy language, there is a strong focus on the Bank's mandate – "core inflation", "cpi inflation"; international trade – "canadian dollar", "united states"; the resource sector – "oil price"; and measures of economic activity – "economic growth", "business investment", "real gdp".

Table 1: Word statistics (N = 17,316)

	Mean	Min	Max
Word Count	23.112	3	87
Number of Positive Words	0.323	0	5
Number of Negative Words	0.573	0	7
Number of Uncertain Words	0.431	0	7
Number of Strong Words	0.089	0	3
Number of Weak Words	0.153	0	4
Number Constraining Words	0.049	0	3

Table 2: Top five words for each category

Positive	Negative	Strong Modal	Weak Modal	Uncertainty	Constraining
stronger	*weaker*	*will*	*somewhat*	*could*	*restrain*
greater	*easing*	*strongly*	*may*	*uncertainty*	*restrained*
strength	*downward*	*lowest*	*suggest*	*somewhat*	*restrictions*
despite	*weak*	*highest*	*suggests*	*risks*	*required*

[19] The WordCloud Python package was used to create figure 3, https://amueller.github.io/word_cloud/.

Figure 4: Monetary policy report word cloud

Notes: A word cloud created by the WordCloud Python package. Larger words are words which appear more frequently in the monetary text.

5.1.2 Reporting similarities – Word Mover's Distance

Another important aspect to look at with monetary policy reports is how they have changed over time. To undergo this task, we will use the *Word Mover's Distance (WMD)* metric, a measure based on the *Earth Mover's Distance (EMD)*. Intuitively, EMD can be thought of as the minimum effort required to move piles of dirt (or earth) to a region of holes, where effort can be understood as the amount of dirt being moved times the distance moved (Rubner et al., 2000). WMD follows the same logic, but instead of measuring the distance between dirt and holes, we are looking at the distance between word embeddings (Kusner et al., 2015) (appendix C). Using this metric, we can look at the distance, or similarity, between two pairs of documents, in our case MPRs (figure 3).

To see how the language in MPRs may have changed, we group a year's worth of monetary reports and compare them with other years from different periods of time. In table 3, *pre-crisis* includes years leading up to 2008, *financial crisis* includes 2008 to 2009, and *post crisis* include all years after 2009 but no later than 2019, excluding the years of the pandemic. As we can see there has been a noticeable shift – increase in distance – in the language used in the MPRs since the financial crisis (the last row). This shift may be explained by the increased emphasis central banks now put on transparency and forward guidance given an increased reliance on unconventional policy (Bernanke, 2022; Dincer et al., 2022). This pattern in behaviour was also evident from the 1990s to 2000s, when the Bank of Canada went from reporting semi-annually to quarterly. The

noticeable shift in language across Canadian monetary reports were also found and referred to as *lexical innovation* from Binette and Tchebotarev (2019).

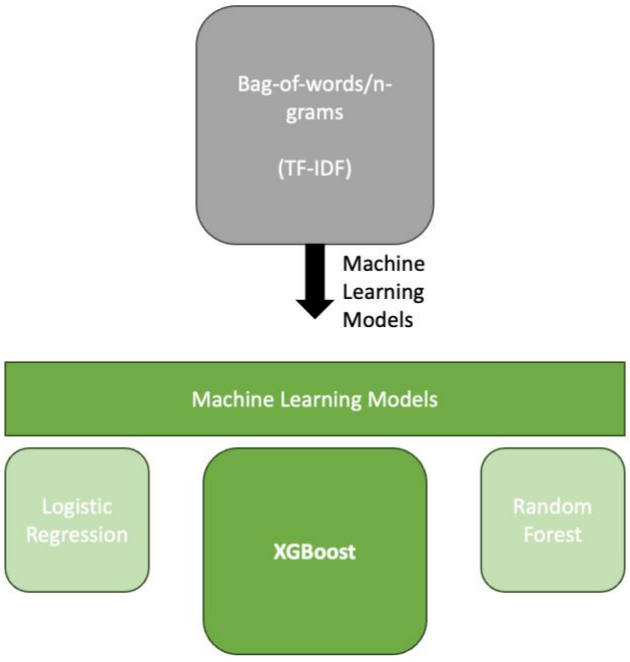
Table 3: WMD across MPRs

	WMD
pre-crisis to pre-crisis	0.27
pre-crisis to financial crisis	0.31
post-crisis to post-crisis	0.28
financial crisis to post-crisis	0.36
pre-crisis to post-crisis	0.38

Notes: *Pre-crisis, financial crisis,* and *post-crisis* refer to the time periods, 2000 to 2007, 2008 to 2009, and 2009 to 2019 respectively. Rows 1 and 2 show the distance between reports within the same period, but of different years. The distance measures are not standardized.

6. Supervised Learning – machine learning models

Figure 5: Machine learning flowchart



To complete the other half of the hybrid method, we have the machine learning models. This paper takes advantage of three machine learning models, but the focus is placed on the high-performance model, *XGBoost*.[20] XGBoost uses both text and numerical features, created by the dictionary model

[20] Other machine learning models – Logistic regression and Random Forest – were also tested and available in the appendix J.

(table 1), to assess sentiment. This model – and subsequent models – will produce an output vector with elements equal to 0, 1, or 2 (negative, neutral, or positive) for sentence level sentiment.

6.1 Primer to supervised-learning

Given the unorthodox methods used, a brief primer may be beneficial. Similar to regression models used in econometrics, supervised machine learning follows a similar path. Many of the terminologies between the two fields will differ but the underlying ideas remain the same. For example, in econometrics a regression can include a *continuous* or *binary* dependent variable. In machine learning, regression refers only to the former, with classification used for the latter. Therefore, this paper is taking advantage of classification modelling.

A large difference between typical econometric modelling and machine learning has to do with *training* and *testing* data. In econometrics, a model is usually fitted onto the whole sample, whereas, in machine learning the model is fitted only to the training sample and then tested on the testing sample. In econometric language, this is essentially *out-of-sample prediction*. By enforcing a "flash card" like learning process on the model, we are forcing the model to learn the underlying characteristics of the data, or the *data generating process*. This training and testing procedure leads to the *bias-variance trade-off* (James et al., 2021). A model with high bias is a model which simplifies the relationship between dependent and independent variables (e.g., a linear model), whereas a model with high variance, is one that complicates the relationship (e.g., a high order polynomial model). High bias leads to a model that is more generalizable to out-of-sample data (testing data), but in turn is less accurate – commonly called *underfitting*.[21] High variance has the opposite problem, not very generalizable but can be extremely accurate on in-sample data (training data) – commonly referred to as *overfitting*. Therefore, a well performing model is one that not only fits well on the training data but also remains accurate on the testing data, which requires a delicate balance of both bias and variance. To train a model that doesn't under- or over-fit, a method called *cross-validation (CV)* is used (James et al., 2021) (appendix D).

[21] Here generalizability refers to how the performance of the model changes to changes in the input data. A model that doesn't generalize well is one that tends to have large fluctuations in performance given small changes in the data.

6.2 Primer to boosted trees

The non-parametric nature of tree-based models has led to a growing interest them amongst economists (Athey & Imbens, 2019) and they are a natural candidate for sentiment analysis.[22] Moreover, of the three machine learning models tested in this study, XGBoost – a boosted tree model – had the best performance.

In essence, boosted models create trees which learn from the mistakes of the past generation of trees (for more see appendix E). The key to boosted trees is that although each tree on its own isn't very powerful (called, a *weak learner*), they become very powerful when combined (Friedman, 2001; Chen, 2014). XGBoost (which stands for Extreme Gradient Boosting) is then an extension of the original boosting method. The main addition that XGBoost brings to the table is scalability and *regularization*.[23] Moreover, the scalability aspect of XGBoost makes it easy to implement given its well implemented Python package.

6.3 Implementing XGBoost

For brevity only the implementation of the XGBoost model will be discussed in detail, though the application of the other models – logistic regression and random forest – followed similar steps.

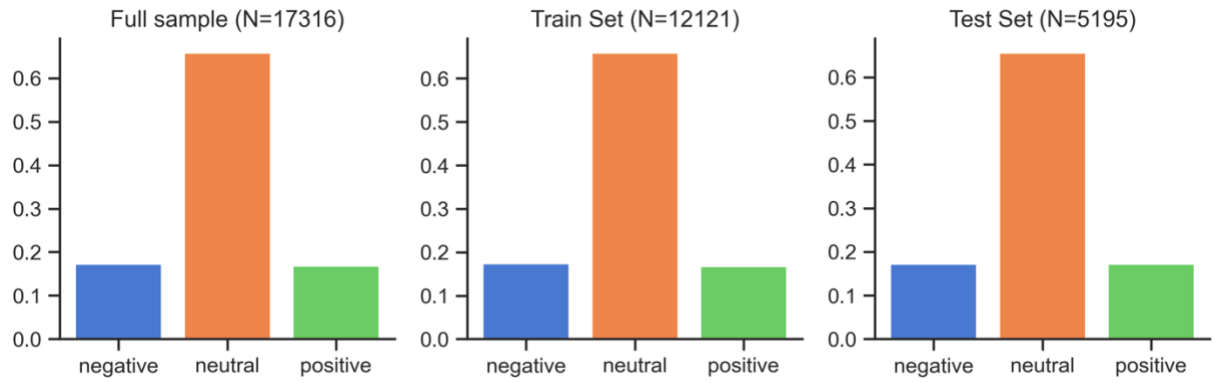
6.3.1 Train-test splits and target imbalance

To train the XGBoost model, the full sample is split into a training and testing set, following the standard guidelines of a 70/30 split – 70 percent for training, 30 percent for testing. Each set is a tabular dataset that contains both the individual sentences, from each MPR, in addition to the numerical terms created by the dictionary method from earlier. This results in a $12{,}121 \times 10$ input matrix used for training; 1 text column plus 9 numeric columns. The target (i.e., dependent variable) is also split in training and testing sets. Importantly, notice that the target is not balanced between the three sentiment classes – over 60 percent of the classified sentences were neutral. This will affect how model performance is measured and is discussed in the next section.

[22] The most basic unit of a tree-based model is known as a decision tree. The specifics of decision trees will be left to the reader to understand but, fundamentally, a decision tree can be thought of as a flow-chart like structure, where outputs are determined by answering a chain of "yes" and "no" questions. A decision tree can also be thought of as a linear regression with many interaction terms, where the interaction term is a split (or decision) at each point in the tree.

[23] Regularization is a method to reduce over-fitting, something that trees are prone to do given their non-parametric nature.

Chart 1: Target (dependent variable) balance



Notes: "Target" is machine learning terminology for "dependent variable". In this case, our target is the sentiment label (negative, neutral, or positive). The vertical axis shows the percentage of the sample size (N) that belongs to each target label.

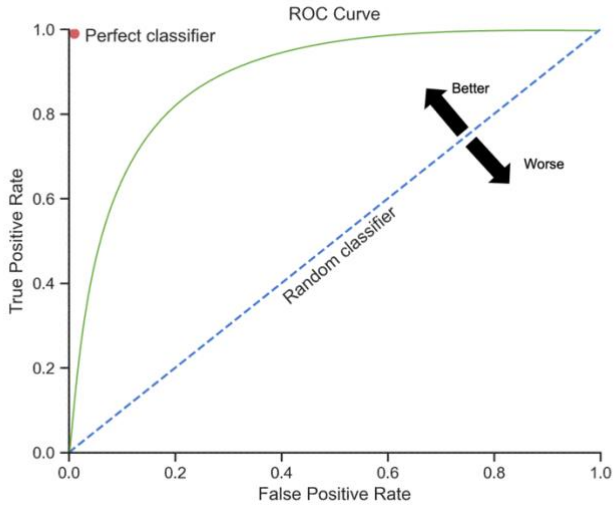
6.3.2 Model metrics

In order to check on how the model is performing both on the test and validation samples, we need a host of metrics. The most obvious way to record model performance is accuracy – the number of true cases divided by the total number of cases. However, given the imbalance of the data set, accuracy would provide overly confident results (Grandini et al., 2020). For example, a model which only predicted the dominant class (neutral) would achieve an accuracy of 66 percent.

Therefore, Performance metrics that should garner more weight are *F1-score* and the *Area Under the Curve (AUC)*.[24] The F1-score is a measurement that naturally deals with imbalanced target data by balancing two classification metrics, *precision* and *recall* (see appendix F). AUC is another useful metric that takes advantage of the *Receiver Operator Characteristic Curve (ROC)* (figure 6). The ROC curve can be thought of as the *power* of the model – the model's ability to correctly reject the null hypothesis – and the AUC is the area underneath it (appendix F). Similar to how the F1-score balances precision and recall, the AUC summarizes the ROC curve.

[24] AUC scores are the calculated area under the Receiver Operator Characteristic (ROC) curve.

Figure 6: ROC curve



Notes: The top left-hand corner is a perfect classifier; 100 percent TP, 0 percent FP. An ROC at the 45-degree line is as good as guessing. Above the 45-degree line is a better classifier and below is worse. The green line represents a theoretical ROC curve for a model.

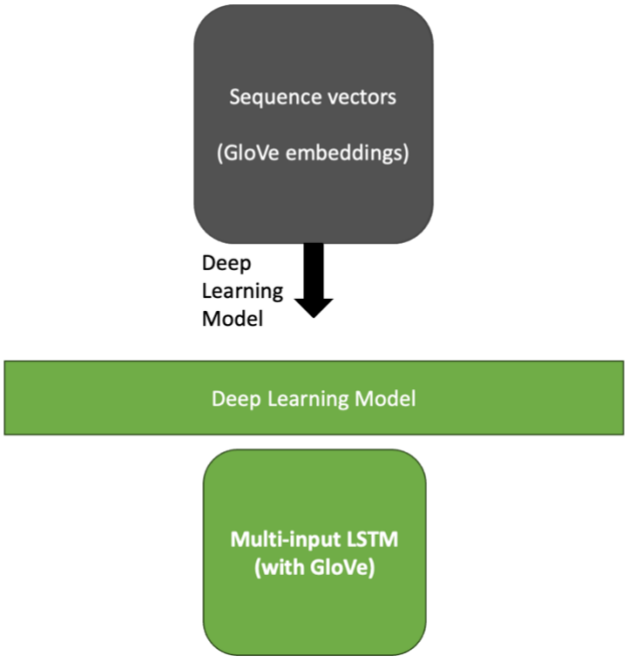
6.3.3 Training XGBoost

An important difference between training – or implementing – a model like XGBoost versus a standard econometric model (e.g., OLS), is the presence of *hyperparameters*. These parameters are ones that are not learned through the model's optimization process – like with beta coefficients – but are instead chosen by the researcher. Therefore, machine learning models often take advantage of hyperparameters to improve fit and predication accuracy. Although different models have different hyperparameters the consequence remains the same, to balance the bias-variance trade-off. Since hyperparameters are chosen by the researcher a form of "trial and error" is required to find an optimal set of parameters for XGBoost (appendix G). The hyperparameters from the highest scoring iteration will be used on the test data.

7. Supervised learning – deep learning model

The following supervised learning model is a multi-input neural network, a deep learning model. The multi-input model consists of a *Long Short-Term Memory (LSTM)* architecture for text features – specifically a sequence of GloVe embeddings – and a *Multilayer Perceptron (MLP)* for numerical features.

Figure 7: Deep learning flowchart



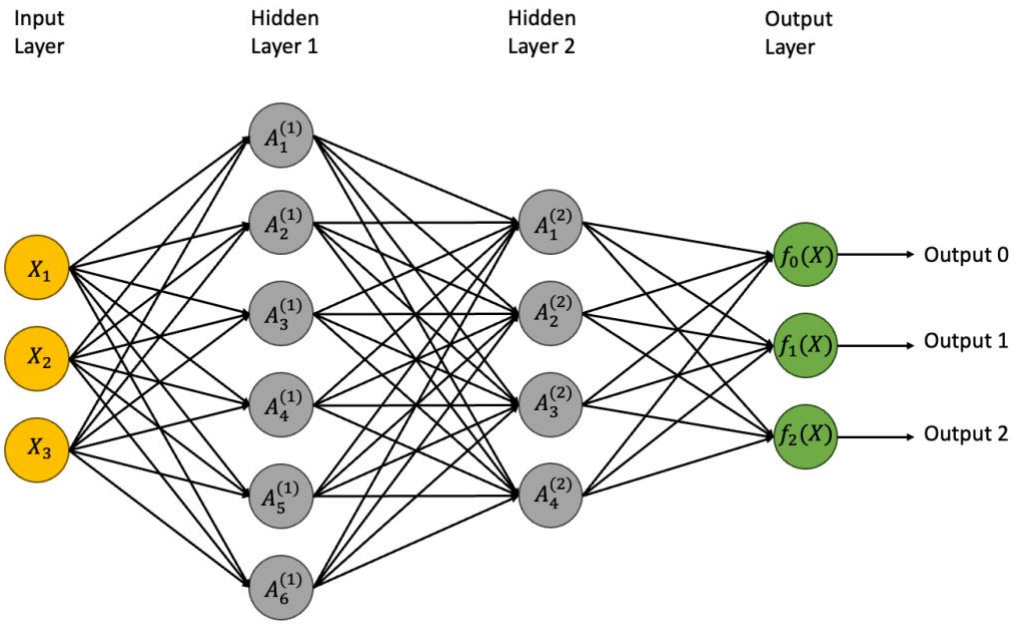
7.1 Neural network primer

Due to the breadth and complexity of neural networks only a high-level overview, specific to the models used, will be provided. Fundamentally, NNs are an artificial construction of the human brain, where an input is received and interpreted by a network of neurons and then aggregated together to form some output.[25] The most basic form of a NN is the MLP model, also known as *feed-forward* models. MLPs consist of a basic structure, an *input layer* of, followed by one or more *hidden layers*, and a final *output layer* (figure 8). Within each layer there are multiple nodes (neurons) all of which are connected to nodes in the next layer, forming a network. For the input and output layers, the nodes correspond to the input features, X, and output functions, $f(X)$, respectively. The output function used in the output layer depends on the problem at hand, in our case we used a *softmax* function.[26] The hidden layers consist of *activation functions*, $A_k^{(i)}$, which is the most unique feature of MLPs. Each non-linear activation function, usually a *ReLu* function, works together to transform the inputs allowing MLPs to effectively model highly non-linear data (appendix H).

[25] Depending on the nature of the task, the number of inputs and outputs can vary. Inputs can be thought of as the predictors or features of the model. Outputs are the predictions.

[26] Softmax is a modified sigmoid function for multiclass cases.

Figure 8: Multilayer perceptron model



Notes: The following figure shows a toy example of a 2-layer MLP model. The nodes of the input layer are in yellow; the hidden layer nodes are in grey, and the output nodes are in green. In reality, the number of hidden layers and nodes per layer are much larger than shown. Every node from a preceding layer has a connection to each node in the next layer, together it creates a network of nodes (or neurons) hence the name, neural network. Re-created and adapted from James et al. (2021).

7.2 LSTM primer

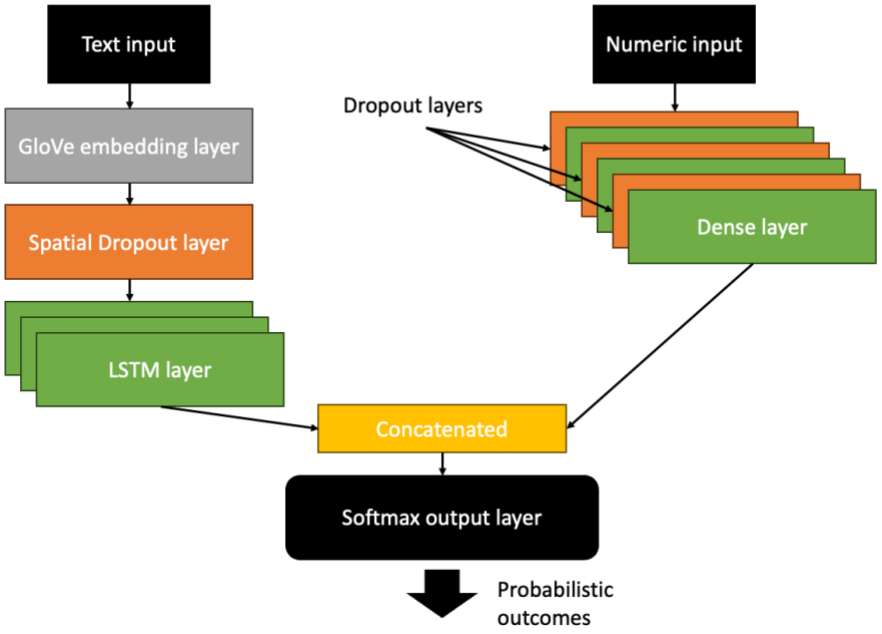
The LSTM, first introduced by Hochreiter & Schmidhuber (1997), is a popular model in natural language processing given its ability to capture sequences – the order of words, or context – with a short- and long-term memory track. The idea of "memory" originates from *Recurrent Neural Networks (RNNs)*, a sequential with only a long-term memory track (Olah, 2015; Challot, 2017; James et al., 2021). A fundamental issue with RNNs is that early inputs get "washed out" when the input sequence becomes too long, much like a long sentence (James et al., 2021). Therefore, LSTMs solve this issue by running both long- and short-term memory pathways throughout the model. The long-term memory track – called the *cell state* – retains all the information that the short-term memory track chooses to keep. The short-term track runs through a series of modules – sets of activation functions – which pick and choose how much of the old information to remember and

how much new information to consider.[27] The output from the short-term modules is remembered by the cell state, representing the model's "long-term memory".

7.3 Implementing a multi-input LSTM with GloVe embeddings

The multi-input LSTM model can be understood as two channels of inputs that feed into two NNs (figure 9), a LSTM for text inputs and a MLP model for numeric inputs. In the final layer, both text and numeric features it will be concatenated together and fed into a softmax layer – producing three probabilities, one for each sentiment class. Finally, an argmax function determines sentiment by selecting the class with the highest probability.

Figure 9: Multi-input LSTM model architecture



Notes: The flowchart shows the architecture of the multi-input LSTM used in this portion of the study. The left-hand side takes care of the text data, and the right-hand side learns the numeric data. The inputs from both channels will become concatenated together and fed into the final softmax layer, which produces the probability that a sentence belongs to each sentiment category. An argmax function is used to determine the predicted sentiment.

[27] The details of about the modules inside of a LSTM are outside of the scope of this paper, however more information on this can be found here: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

The architecture of the text channel (the LSTM) follows closely to that used by Vernikou et al. (2022) on twitter data. The first layer from the text channel is a pre-trained GloVe[28] word embedding, created by Pennington et al. (2014). This layer fits the words in the MPRs to the pre-trained embedding space, converting over 5000 unique words.[29] The numeric channel, follows a repeated pattern of three *dropout* and three *dense* – or feed-forward – layers, forming a MLP model. Both channels include a dropout layer, a form of regularization commonly used in the deep learning space (appendix I).

8. Results of NLP models

Table 4: XGBoost Classification Report

	Negative	Neutral	Positive	Accuracy	Macro average	Weighted average	AUC
Precision	0.835	0.789	0.740	-	0.788	0.788	-
Recall	0.443	0.941	0.543	-	0.642	0.787	-
F1-score	0.579	0.858	0.626	-	0.688	0.770	-
Support	893	3408	894	0.787	5195	5195	0.876

Notes: Negative, Neutral, and Positive are the sentiment categories. Each row shows the individual measure (e.g., Precision) for each category. Macro average considers all three classes but is unweighted. Weighted average accounts for imbalance. Accuracy is not adjusted for target imbalance, but AUC is – weighted average. Support displays the number of observations by category. All results are on the test sample.

Table 5: Multi-input LSTM Classification Report

	Negative	Neutral	Positive	Accuracy	Macro average	Weighted average	AUC
Precision	0.758	0.797	0.759	-	0.771	0.784	-
Recall	0.501	0.933	0.521	-	0.652	0.788	-
F1-score	0.603	0.860	0.618	-	0.694	0.774	-
Support	893	3408	894	0.788	5195	5195	0.877

Notes: refer to table 4.

After training and validation, each model is tested on the test sample – MPR sentences which the models have yet to see. Once the test data is fed into the model, a sentiment class is returned, resulting in a vector with elements equal to 0, 1, or 2. Tables 4 and 5 show the test performance of XGBoost and the LSTM across multiple metrics. The support row shows that observations used for both models are equal. The Negative, Neutral, and Positive columns represent the measures for each individual class. In contrast Accuracy, is rated across all classes; macro average and weighted average measures (includes AUC) are adjusted for class prevalence (table 4 notes).

[28] This pre-trained layer was trained by the Stanford team on *Wikipedia* and *Gigaword 5* text data, producing 6 billion tokens and a vocabulary of 400,000 words.

[29] 686 words were not embedded, meaning that they are out-of-vocabulary (OOV). OOVs are left as vectors of zeros. Each vector in or out of vocabulary is padded to a common length to ensure that the dimensions are consistent.

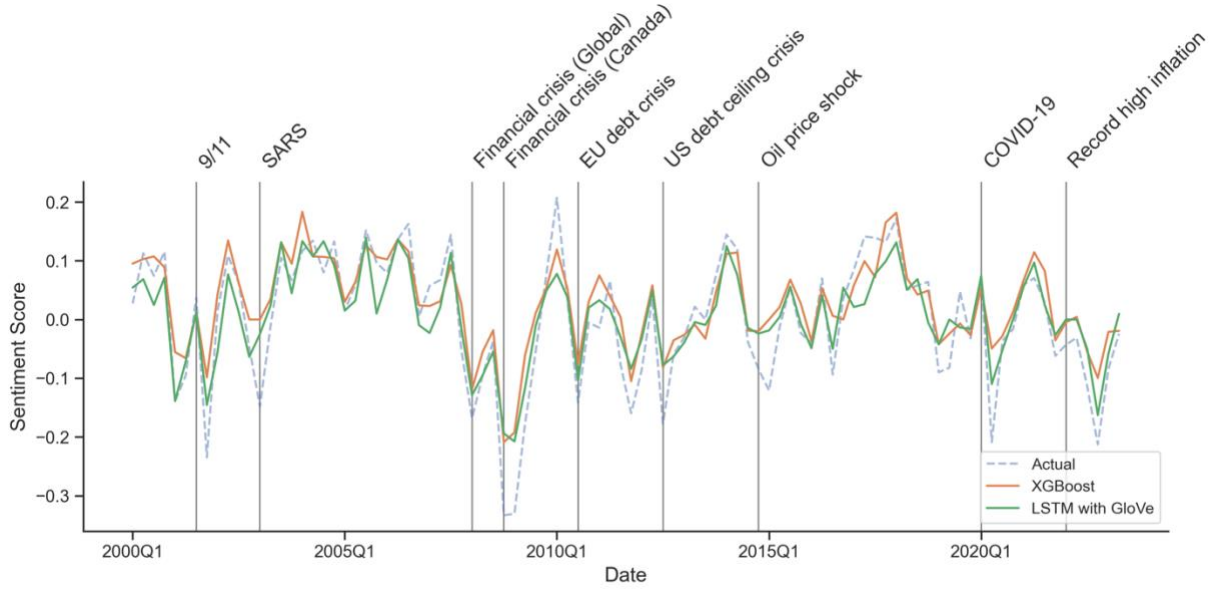
From testing we find that the LSTM consistently outperforms the XGBoost model, across: Accuracy (0.788 > 0.787), weighted F1-score (0.774 > 0.770), and weighted AUC (0.877 > 0.876). Given these results, the predicted sentiment from the LSTM will be used in the forecasting model.

9. Forecasting – effects of inside sentiment on interest rates

Given the resulting sentiment feature from the NLP models, we can further investigate whether sentiment influences the change in interest rates (i.e., policy decisions). A first order autoregressive – or AR(1) – model, via OLS estimation, is employed to track the relationship between sentiment and interest rates changes.[30] Three measures of interest rates will be used, the Bank of Canada's *Canadian Overnight Repo Rate Average (CORRA)*, the prime interest rate, and mortgage rates[31]. Prime and mortgage rates were included as they have greater variability – during stable financial periods – in addition to being more representative of the effects that sentiment may have on the wider economy. Other macroeconomic variables – *CPI*, *Canadian effective exchange rates*, *real GDP*, and *unemployment* – for robustness.

9.1 Sentiment metric and time series data

Figure 10: Sentiment predictions across time



Notes: Sentiment metric plotted in chronological order. "Actual" is the result from human interpretation of sentiment (i.e., sentiment graded), "XGBoost" and "LSTM with GloVe" are the predictions from the trained models, as shown in section 8.

[30] An econometric model allows for the calculation of standard errors and inference.

[31] 1-year, 3-year, and 5-year mortgage rates.

To create the sentiment metric, sentence level sentiment is aggregated by MPR and normalized. Therefore, the difference between positive and negative sentences is divided by the total number of sentences, per MPR:

$$sentiment\ metric = \frac{\#\ positive - \#\ negative}{\#\ positive + \#\ negative + \#\ neutral}$$

Plotting the sentiment metric – the model predicted sentiment and human classified sentiment (i.e., "Actual") – over time shows a pattern similar to that found by Binette and Tchebotarev (2019). We can observe the major troughs in sentiment during notable macro-events, the 9/11 attacks, the global financial crisis, COVID-19, and the record high inflation seen shortly after the start of the pandemic. Note that both the global impact and the Canadian impact of the financial crisis were marked in figure 10; Canada was one of the last countries to experience a downturn. Interestingly, we can see that the downturn in sentiment from COVID was not as deep as for the financial crisis and even recovered quickly. This falls in line with the impact that extraordinary monetary stimulus had on financial markets. Figure 10 also extends the results seen from section 8, showing the "fit" of the LSTM on the data.

Although, the human classified sentiment could have been used for forecasting, the purpose here is to show the capability of supervised learning models and, in turn, provide evidence that an efficient data pipeline can be constructed.

The remaining macro-variables are cleaned, averaged, and differenced. Summary statistics are available in table 6.

Table 6: Time series summary statistics (N = 94)

	Mean	Std	Min	Max
CORRA	-0.002	0.417	-1.364	1.508
Mortgage (1 Year)	-0.009	0.373	-1.442	1.492
Mortgage (5 Year)	-0.018	0.266	-1.138	0.854
Mortgage (3 Year)	-0.019	0.343	-1.401	1.158
Prime	0.004	0.408	-1.385	1.442
Sentiment	0.006	0.075	-0.208	0.139
CPI	0.667	0.828	-1.733	4.533
Unemployment	-0.019	0.845	-3.233	6.833
Real GDP	0.54	1.979	-14.015	10.301
Exchange rate	0.179	3.273	-12.798	7.133

Notes: All variables are first differenced (i.e., $x_t - x_{t-1}$), as unit changes, and converted to quarterly frequency. Interest rate measures and unemployment are percentages (i.e., percentage point change). Sentiment is our calculated and normalized metric – sentiment by MPR. CPI, real GDP, and exchange rates are indices (i.e., change in index). The GDP index was created with quarterly data, base year is 2005.

9.2 Forecasting model

The dynamic empirical model used for the forecasting exercise will follow the basic form:

$$y_t = \beta_0 + \gamma Sentiment_t + \beta_1 y_{t-1} + \beta_2 CPI_{t-1} + \beta_3 Unemployment_{t-1} + \beta_4 GDP_{t-1} + \beta_5 Exchange_{t-1} + \epsilon_t \quad (1)$$

where, y_t is one of three interest rates variables (CORRA, mortgage rates, and prime rates) and y_{t-1} represents the lagged value. $Sentiment_t$ is our sentiment metric and the remaining variables are macroeconomic controls, all differenced and lagged one period. Unlike the controls, our sentiment is not explicitly lagged in the model's specification. This was intentionally done, as the information lag present when writing the MPRs implicitly lags sentiment by at least one period. Therefore, lagging sentiment within the model could induce a two-period lag, 8 months. However, as a robustness check the results of a lagged sentiment is available in the appendix K.

9.3 Forecasting results

In the results, with CORRA as the dependent variable (table 7), we can see that sentiment has a strong and significant relationship with changes in CORRA for all three specifications. Unsurprisingly, the effect of sentiment is largest when no other controls are included (column 1), and weakest with all controls (column 3). However, even with a full set of control variables, sentiment has the largest influence on changes in CORRA – a result that continues with every tested dependent variable (table 8). The results on mortgage rates tell a similar story, showing that sentiment remains the strongest predictor even on wider reaching interest rates, although this effect is weakest with the 5-year mortgage rate (table 8). Among the macroeconomic controls, CPI and unemployment tend to be the most significant predictors of rate changes; following the Bank of Canada's inflation mandate and widely understood relationship from the Philips curve (Philips, 1958; Corrigan et al., 2021). The weak autoregressive relationship with 3-year mortgages is also captured and explained by the weak nature that a one quarter lag would have on a longer-term indicator. This relationship does not hold true with 1-year mortgage rates.

Table 7: CORRA results

	(1)	(2)	(3)
R-squared	0.1146	0.4821	0.6377
R-squared Adj.	0.1049	0.4706	0.6124
const	-0.0122	-0.0093	-0.1864***
	(0.0438)	(0.0329)	(0.0503)
Sentiment	1.8697**	1.2943**	1.0923**
	(0.8391)	(0.5683)	(0.5256)
CORRA (t-1)		0.6152***	0.6162***
		(0.1233)	(0.0812)
CPI			0.1866***

	(0.0474)
Exchange rate	0.0042
	(0.0078)
Real GDP	0.1196**
	(0.0521)
Unemployment	0.3848***
	(0.1209)

Standard errors in parentheses.

* p<.1, ** p<.05, ***p<.01

Notes: Column 1 includes only the sentiment indicator. Column 2 adds the lagged dependent. Column 3 includes all other macroeconomic controls.

Table 8: All results with sentiment

	CORRA	Prime rate	Mortgage 1-year	Mortgage 3-year	Mortgage 5-year
R-squared	0.6377	0.6192	0.5124	0.424	0.3458
R-squared Adj.	0.6124	0.5926	0.4784	0.3839	0.3002
const	-0.1864***	-0.1798***	-0.1747***	-0.1874***	-0.1528***
	(0.0503)	(0.0524)	(0.0481)	(0.054)	(0.0422)
Sentiment	1.0923**	1.0473**	1.1044**	1.2886***	0.7276*
	(0.5256)	(0.5124)	(0.5392)	(0.4912)	(0.393)
CORRA (t-1)	0.6162***				
	(0.0812)				
Prime rate (t-1)		0.6176***			
		(0.0847)			
Mortgage (1-year) (t-1)			0.4233***		
			(0.1155)		
Mortgage (3-year) (t-1)				0.1617	
				(0.1176)	
Mortgage (5-year) (t-1)					0.0525
					(0.1083)
CPI	0.1866***	0.1763***	0.1650***	0.1911***	0.1365***
	(0.0474)	(0.0469)	(0.0577)	(0.0502)	(0.0398)
Exchange rate	0.0042	0.0039	0.0061	0.0104	0.0160**
	(0.0078)	(0.0076)	(0.0078)	(0.0089)	(0.0081)
Real GDP	0.1196**	0.1234**	0.1084**	0.0743	0.0769**
	(0.0521)	(0.0541)	(0.0458)	(0.0495)	(0.0356)
Unemployment	0.3848***	0.3860***	0.2714***	0.1885*	0.1897**
	(0.1209)	(0.1262)	(0.1046)	(0.1112)	(0.0797)

Standard errors in parentheses.

* p<.1, ** p<.05, ***p<.01

Notes: Each column shows the results for a given interest rate series with full specification; the lagged dependent and all macroeconomic controls are included.

More importantly, the results show that text can provide a window into the sentiment held by policy makers, and further supports the significance of managing expectations, something that standard macro-indicators fail to do on their own. In other words, sentiment – from MPRs –

includes valuable information on interest rate changes which are not offered by standard macro-indicators. To show this more concretely we run an exercise similar to that by Romer and Romer (2004). By regressing sentiment on the macroeconomic controls and obtaining the residuals, we can effectively extract the additional information that sentiment offers, without influence from other indicators. Table 9 displays the lack lustered fit (*R-squared Adj. = 0.1*) of the model (equation

$$Sentiment_t = \alpha_0 + \alpha_1 CPI_{t-1} + \alpha_2 Unemployment_{t-1} + \alpha_3 GDP_{t-1} + \alpha_4 Exchange_{t-1} + e_t \quad (2)$$

Table 9: Macroeconomic sentiment

	(1)
R-squared	0.1389
R-squared Adj.	0.0998
const	-0.0094
	(0.0127)
CPI	0.0038
	(0.0097)
Exchange Rate	0.0066**
	(0.0027)
Real GDP	0.0220*
	(0.0132)
Unemployment	0.039
	(0.0274)

Standard errors in parentheses.

* p<.1, ** p<.05, ***p<.01

Notes: These results show the predictive power of our macro-indicators on sentiment (i.e., sentiment as the dependent variable). The low R-squared Adj. shows that our standard indicators are not able to explain sentiment very well. This result strongly suggest that sentiment is offering additional variation which standard indicators fail to introduce.

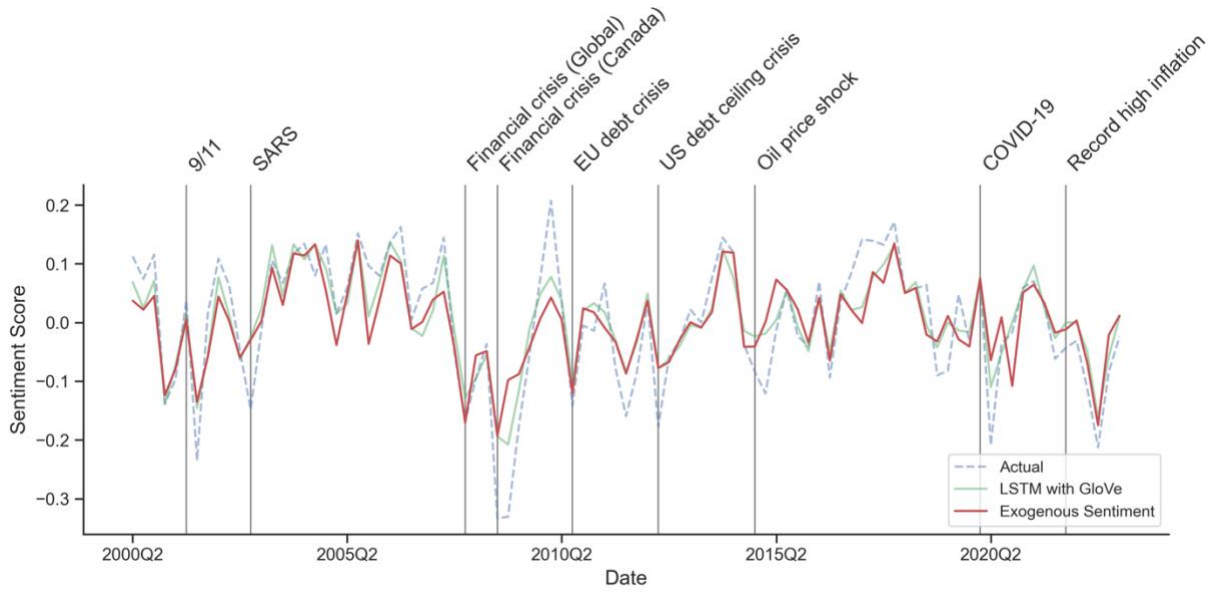
2) and figure 11 visualizes the residuals. This result shows variation in sentiment which is orthogonal to macro-indicators (i.e., 'Exogenous Sentiment'), remains similar to observed sentiment (i.e., 'Actual'). Therefore, Romer and Romer's (2004) seminal idea of "monetary policy shocks"[32] can, equivalently, be expressed as changes in sentiment independent of the economic state.

As a result, our models show that inside sentiment has the highest predictive power – above all other controls – and provides novel variation (table 9 & figure 11) which is not possible with these standard measures alone. As a result of the added information set, our models are able to

[32] Monetary policy shocks are defined as monetary decision exogenous of Greenbook forecasts (Romer & Romer, 2004; see section 1.1).

anticipate policy changes and changes in broader interest rates to a greater degree than with standard specifications (see appendix K for additional results).

Figure 11: Exogenous sentiment



Notes: Refer to figure 10. "Exogenous Sentiment" is the residual from the model in table 9. This series represents the effects of sentiment not picked up by our macroeconomic controls. We can see that the exogenous series is still similar to the graded series, 'Actual'.

10. Limitations and future work

Although our multi-input LSTM provides some impressive results it remains a fairly naïve model due to its inherently low complexity. As an example, our LSTM has 1.3 million parameters – of which 115,000 are trainable[33] – compared to a "basic" *Bidirectional Encoder Representations from Transformers* (BERT) model which hosts 110 million parameters. [34] Additionally, a simpler model was necessary due to the limited resources available, such as hardware, training time, and the size of the dataset. State-of-the-art transformers, like BERT, require enormous amounts of processing power and extraordinary amounts of data (Kotei & Thirunavukarasu, 2023). A bypass to this is to use transfer learning. However, even this requires a considerable amount of processing power to fine-tune and run inference[35] (i.e., testing).

[33] Only a fraction of all parameters is "trainable" because of the pre-trained GloVe embedding.

[34] This is according to Hugging Face (a python package for transformers) and their "base" BERT model; https://huggingface.co/transformers/v3.3.1/pretrained_models.html.

[35] Note that inference in the AI space does not mean statistical inference. Here, inference refers to using a trained model on out-of-sample data.

Moreover, our models require all observations to be labelled (as mentioned in section 3.1), whereas models like BERT can perform most of their learning unsupervised, requiring fewer labelled observations for fine-tuning (Devlin et al., 2019); the same applies with transfer learning. An obvious issue (discussed in section 3.1) with grading is the accuracy of human graded sentiment. Without a substantial amount of time and several independent auditors grading is more prone to error. Lastly, as with most of the other limitations, due to time constraints, less optimal models were considered for the forecasting procedure (i.e., OLS estimation).

Therefore, future work should focus on using transformer models and techniques like transfer learning to fine-tune highly accurate models that require less labelled data and less training time. In addition, more suitable time series models and estimation methods should be considered for higher quality inference.

11. Conclusion

The work in this study provides evidence of three main points. First it shows that NLP and supervised learning models can be harnessed to effectively learn and predict sentiment from financial and economic text (i.e., MPRs). Second, the results of such a model (i.e., the sentiment metric) can be used as a predictor for changes in policy rates – and wider reaching interest rates (e.g., mortgage rates) – by providing novel information and consequently, more predictive power than standard macroeconomic indicators. Third, the methods used provide a blueprint for the implementation of text as a feature in subsequent economic applications (e.g., forecasting), and significantly improve the speed at which such a task can be completed. Specifically, the data pipeline from this study, allows for sentiment to be extracted from subsequent editions of the MPR, without further training.

Future works should look to apply more advanced deep learning methods; to further improve accuracy and reduce training time for sentiment analysis. A similar measure of sentiment could then be constructed and used to predict other macroeconomic indicators.

References

Aruoba, B. (2022). Identifying Monetary Policy Shocks: A Natural Language Approach. *CEPR Discussion Papers*. https://ideas.repec.org/p/cpr/ceprdp/17133.html

Athey, S., & Imbens, G. (2019). *Machine Learning Methods That Economists Should Know About*. https://doi.org/10.1146/annurev-economics-

Bernanke, B. S. (2022). 21st Century Monetary Policy: The Federal Reserve from the Great Inflation to COVID-19. In *Google Books*. W. W. Norton & Company. https://books.google.ca/books?id=qAJLEAAAQBAJ&printsec=frontcover#v=onepage&q&f=false

Binette, A., & Tchebotarev, D. (2019, February 6). *Canada's Monetary Policy Report: If Text Could Speak, What Would It Say?* Www.bankofcanada.ca. https://www.bankofcanada.ca/2019/02/staff-analytical-note-2019-5/

Bird, S., Klein, E., & Loper, E. (2019). *NLTK Book*. Nltk.org. https://www.nltk.org/book/

Blinder, A., Ehrmann, M., Fratzscher, M., De, J., & Jansen, D.-J. (2008). *Central Bank Communication and Monetary Policy Report a Survey of Theory and Evidence*. https://www.ecb.europa.eu/pub/pdf/scpwps/ecbwp898.pdf

Böök, A., Jurado, J., & Palome, D. F. (2019, July 6). *Using sentiment analysis to capture monetary policy decisions*. Medium. https://inside-techlabs.medium.com/using-sentiment-analysis-to-capture-monetary-policy-decisions-bfe3d9a13857

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/a:1010933404324

Chen, T. (2014). *Introduction to Boosted Trees*. https://web.njit.edu/~usman/courses/cs675_fall16/BoostedTree.pdf

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 785–794. https://doi.org/10.1145/2939672.2939785

Chollet, F. (2018). *Deep Learning with Python*. Manning, Cop.

Coase, R. H. (1960). The Problem of Social Cost. *The Journal of Law & Economics*, *3*, 1–44. https://www.jstor.org/stable/724810

Consoli, S., Barbaglia, L., & Manzan, S. (2022). Fine-grained, aspect-based sentiment analysis on economic and financial lexicon. *Knowledge-Based Systems*, *247*, 108781. https://doi.org/10.1016/j.knosys.2022.108781

Corrigan, P., Desgagnés, H., Dorich, J., Lepetyuk, V., Miyamoto, W., & Zhang, Y. (2021, June 28). *ToTEM III: The Bank of Canada's Main DSGE Model for Projection and Policy Analysis*. Www.bankofcanada.ca. https://www.bankofcanada.ca/2021/06/technical-report-119/#:~:text=The%20Terms%2Dof%2DTrade%20Economic

Devlin, J., Chang, M.-W., Lee, K., Google, K., & Language, A. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.

Dincer, N., Eichengreen, B., & Geraats, P. (2022). *Trends in Monetary Policy Transparency: Further Updates*. https://www.ijcb.org/journal/ijcb22q1a8.pdf

DP18125 Text Algorithms in Economics. (2023, April 28). CEPR. https://cepr.org/publications/dp18125

Frankel, R., Jennings, J., & Lee, J. (2021). Disclosure Sentiment: Machine Learning vs. Dictionary Methods. *Management Science*. https://doi.org/10.1287/mnsc.2021.4156

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, *29*(5), 1189–1232. https://www.jstor.org/stable/2699986

FRIEDMAN, M., & SCHWARTZ, A. J. (1963). A Monetary History of the United States, 1867-1960. In *JSTOR*. Princeton University Press. https://www.jstor.org/stable/j.ctt7s1vp

Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for Multi-Class Classification: an Overview. *ArXiv:2008.05756 [Cs, Stat]*. https://arxiv.org/abs/2008.05756

HaCohen-Kerner, Y., Miller, D., & Yigal, Y. (2020). The influence of preprocessing on text classification using a bag-of-words representation. *PLOS ONE*, *15*(5), e0232525. https://doi.org/10.1371/journal.pone.0232525

Hastie, T., Tibshirani, R., & Friedman, J. (2008). *Springer Series in Statistics The Elements of Statistical Learning Data Mining, Inference, and Prediction Second Edition*. https://hastie.su.domains/Papers/ESLII.pdf

He, H., & Ma, Y. (2013). Imbalanced Learning. In *Wiley eBooks*. Wiley. https://doi.org/10.1002/9781118646106

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(1), 1–42. https://doi.org/10.1162/neco.1997.9.1.1

Howard, J., & Ruder, S. (2018). *Universal Language Model Fine-tuning for Text Classification*. ArXiv.org. https://arxiv.org/abs/1801.06146

Ioffe, S., & Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. ArXiv.org. https://arxiv.org/abs/1502.03167

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2023). *An Introduction to Statistical Learning with Applications in R Second Edition*. https://hastie.su.domains/ISLR2/ISLRv2_corrected_June_2023.pdf

Jianqiang, Z., & Xiaolin, G. (2017). Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis. *IEEE Access*, *5*, 2870–2879. https://doi.org/10.1109/access.2017.2672677

Kalamara, E., Turrell, A., Redl, C., Kapetanios, G., & Kapadia, S. (2022). Making text count: Economic forecasting using newspaper text. *Journal of Applied Econometrics*. https://doi.org/10.1002/jae.2907

Kaur, J., & Buttar, P. K. (2018). A Systematic Review on Stopword Removal Algorithms. *International Journal on Future Revolution in Computer Science & Communication Engineering*, *4*(4), 207–210. https://www.ijfrcsce.org/index.php/ijfrcsce/article/view/1499

Keynes, J. M. (1936). *The general theory of employment, interest, and money: interest and money.* https://scholar.google.com/scholar?hl=en&as_sdt=0,5&cluster=11264128813459472212#d=gs_cit&t=1689960629811&u=%2Fscholar%3Fq%3Dinfo%3AVH9-1nk5UpwJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26scfhb%3D1%26hl%3Den

Kim, J.-H. (2009). Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics & Data Analysis*, *53*(11), 3735–3745. https://doi.org/10.1016/j.csda.2009.04.009

Kohavi, R. (1995). *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*. https://www.ijcai.org/Proceedings/95-2/Papers/016.pdf

Kotei, E., & Thirunavukarasu, R. (2023). A Systematic Review of Transformer-Based Pre-Trained Language Models through Self-Supervised Learning. *Information*, *14*(3), 187. https://doi.org/10.3390/info14030187

Kowsari, K., Jafari Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text Classification Algorithms: A Survey. *Information*, *10*(4), 150. https://doi.org/10.3390/info10040150

Kusner, M., Sun, Y., Kolkin, N., & Weinberger, K. (2015, June 1). *From Word Embeddings To Document Distances*. Proceedings.mlr.press; PMLR. https://proceedings.mlr.press/v37/kusnerb15.html

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.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Distributed Representations of Words and Phrases and their Compositionality*. https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Olah, C. (2015, August 27). *Understanding LSTM Networks*. Github.io. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.3115/v1/d14-1162

Phillips, A. W. (1958). The Relation between Unemployment and the Rate of Change of Money Wage Rates in the United Kingdom, 1861-1957. *Economica*, *25*(100), 283.

Rong, X. (2014). *word2vec Parameter Learning Explained*. ArXiv.org. https://arxiv.org/abs/1411.2738

Rubner, Y. (2000). The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, *40*(2), 99–121. https://doi.org/10.1023/a:1026543900054

Sanderson, M. (2010). Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. ISBN-13 978-0-521-86571-5, xxi + 482 pages. *Natural Language Engineering*, *16*(1), 100–103. https://doi.org/10.1017/s1351324909005129

Shiller, R. J. (2017). Narrative Economics. *American Economic Review*, *107*(4), 967–1004. https://doi.org/10.1257/aer.107.4.967

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015). Efficient Object Localization Using Convolutional Networks. *ArXiv:1411.4280 [Cs]*. https://arxiv.org/abs/1411.4280

Vernikou, S., Lyras, A., & Kanavos, A. (2022). *Multiclass sentiment analysis on COVID-19-related tweets using deep learning models*. https://doi.org/10.1007/s00521-022-07650-2(

Wankhade, M., Rao, A. C. S., & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, *55*. https://doi.org/10.1007/s10462-022-10144-1

Appendix

A. TF-IDF

The construction of a TF-IDF vector requires two measurements, the term frequency – per document – and the inverse document frequency. Term frequency per document is calculated by counting the total number of times a term *(t)* appears in a document *(d)* and dividing that number by the total number of terms in that document.

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

Since higher term frequency doesn't necessarily mean greater importance, the inverse document frequency is also calculated. The IDF follows the inverted logic of TF, less frequently occurring words are more likely to provide more information than common words. Formally, the IDF is the logarithmically scaled ratio of the total number of documents *(N)* to the number of documents *(D)* term *t* appears in.

$$IDF(t,D) = log\frac{N}{|\{d \in D: t \in d\}|}$$

With both TF and IDF calculated, TF-IDF is simply the product of the two terms:

$$TFIDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

B. Sequence vectors and word embeddings

For sequence vectors similar pre-processing steps – to BoW – are used. The most basic method of vectorizing sequence vectors is with one-hot encoding – as with BoW. However, the resulting one-hot encoded matrix differs from the one seen in section 4.3 (BoW vectorization) in that each row of the matrix represents a word (or token) and the number of rows (length) *n* is equal to the size of the vocabulary. This leads to a major dimensionality issue since each vector represents one word rather than one document, or sentence (James et al., 2021). Therefore, the sentence, *"The cat chased the mouse"*, would return four sparse vectors rather than one.

Word embeddings improve upon one-hot encoding and solve the dimensionality issue by reducing large sparse matrices into smaller dense matrices (Rong, 2016). The key to dense outputs is that word embeddings can capture similarities between pairs of words and implicitly understand semantics (Kowsari et al., 2019). This allows the representative matrix to be denser and thus smaller than one created by one-hot encoding. In essence, word embeddings, namely Word2Vec, work by mapping token vectors to a n-dimensional space, where tokens that are closer, are words

which carry similar meaning.[36] In the case of Word2Vec this task of mapping token vectors to another vector space, is done by a shallow neural network.[37]

C. WMD

WMD applies the idea of minimum work and applies it to the distances between a pair of documents located in the n-dimensional space created by Word2Vec (Kusner et al., 2015). Therefore, instead of moving piles of dirt into holes, we are moving words vectors from one document such that they match the word vectors from another document, within the Word2Vec space.

D. Cross-validation

Rather than splitting the whole sample into one training sample and one testing sample, CV takes the training sample and splits it into *k-folds*, where each split in the data consists of training and *validation* folds. The model is then iteratively trained and validated among each set – or fold – of training and validation samples – usually 5 or 10 times; *k=5* or *k=10* (Kohavi, 1995; Hastie et al., 2009). In other words, rather than training on the training sample and testing on the testing sample, CV trains and validates on multiple subsamples of the initial training sample. This iterative process allows for robust results and in turn better performance, even when compared to similar methods like *bootstrapping* (Kohavi, 1995; Kim, 2009).

E. XGBoost

XGBoost is closely related to Random Forests (Breiman, 2001), which makes use of bootstrap aggregation (*bagging*) to, *independently*, create many trees (i.e., a forest), boosting works by *sequentially* adding trees which are created on the residuals of past trees. Formally, we can see that the objective function of an XGBoost model as:

$$\text{objective} = \sum_{i}^{n} l(y_i, \hat{y_i}) + \sum_{k=1}^{K} \omega(f_k)$$

Similar to OLS, something economists hold near and dear, the objective function is optimized given the loss function $l(y_i, \hat{y_i})$ (e.g., mean squared error) and the, additional,

[36] The output dimension of the word embeddings *n* is chosen by the researcher.

[37] A shallow neural network is a network with 2 layers. Unlike Word2Vec, GloVe leverages statistical methods rather than neural networks to understand similarities between words, Pennington et al. (2014).

regularization term $\omega(f_k)$, where $\omega(*)$ is a function of the complexity of the tree f_k (Chen & Guestrin, 2016).[38] The additive regularization term can be thought of as a constraint in a minimization problem, making the objective function harder to optimize and reducing the chance of overfitting.

F. Metrics – F1-score and AUC

The F1-score measure accounts for data imbalance this by giving a *harmonic mean* of precision and recall (Gradinin et al., 2020).

Formally, recall is:

$$recall = \frac{TP}{TP + FN}$$

where *TP* is the number of *True Positives* and *FN* is a count of all *False Negatives*. In other words, recall tells us how well the model is able to pick out positive classes. Precision can be understood as:

$$precision = \frac{TP}{TP + FP}$$

where *FP* are the number of *False Positives*. Intuitively, precision can be understood as how many of the predicted positive classes were actually positive classes (i.e., true positives). Both precision and recall exclude *True Negatives (TN)* which makes them useful when the target class is imbalanced (He & Ma, 2013). However, precision and recall work in opposites, meaning that a model with high recall must, by definition, have poor precision and vice versa – this relationship is represented by the *precision-recall curve*. Thus, the two metrics can become difficult to interpret.

$$F1 = 2 * \left(\frac{precision * recall}{precision + recall}\right)$$

Specifically, given that sentiment is a multiclass problem, we will be using a *weighted macro average*:[39]

$$F1\ weighted\ average = \frac{1}{\sum_{l \in L} |y_l|} \sum_{l \in L} |y_l| * \text{F1}$$

where L is the set of labels and y_l is the set of targets in label l. The macro average is just the unweighted average among all three classes. Weighted scores are weighted by the number of appearances for each class therefore, classes with more observations do not give disproportionate weight.

[38] The actual inner working of tree complexity is outside of the scope of this paper but is available here, https://xgboost.readthedocs.io/en/stable/tutorials/model.html

[39] F1 weighted average is the F1 score averaged for each class (label), by the number of TP for that class.

The ROC curve (figure 6) relates TP to FP rates, where at each point along the curve represents the TP and FP rates for a given model at a different threshold. Where the threshold is the minimum level at which a predicted class will flip from one to another (i.e., from 0 to 1). Normally, this threshold is 0.5, or 50 percent. The AUC score will also be adjusted for multiclass labels.

G. Training XGBoost

For XGBoost CV was used to tune four hyperparameters are tuned; *number of estimators*, *portion to subsample*, *L2 regularization*, and the *learning rate*. Since tree-based methods are prone to overfitting, most of the hyperparameters are used to increase the bias of the model. Specifically, parameters like number of estimators, control for how deep each tree can expand, where deeper models have higher variance. Subsampling forces the model to use only a subset of the full sample to create each tree (similar to bagging), increasing bias. Like subsampling and the former, L2 regularization works to increase bias, by increasing the strength of the regularization term in the objective function, $\omega(f_k)$. The learning rate is used to control the "step-size" of *gradient descent* – the optimization process (Chollet, 2017; James et al., 2021).[40] The hyperparameter tuning is done by looping through a list of hyperparameters and performing CV for each set of parameters. In addition, *scikit-learn pipelines* were used during the whole of the training process, including the steps to vectorize the text. Pipelines help to both streamline the training process and prevent *data leakage*.[41]

H. Neural networks

Interestingly, a basic single-layer MLP is similar to a linear regression with K regressors:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k$$

Thus, it's the linear combination of non-linear activation functions that make MLPs useful (James et al., 2021).

[40] Gradient descent is the commonly used optimization process to find a local minimum of a given loss function. Unlike with OLS which instantaneously finds the optimal solution, gradient descent does so iteratively. The key strength of gradient descent is that it does not require the model to be linear and is thus a flexible method which can be used for many different models.

[41] Data leakage happens with information from outside the training sample is used during training (e.g., if standardization was used and data outside of the training sample was used to calculate mean and standard deviation).

I. Implementing the multi-input LSTM

Both channels use dropout layers to regularize the model, similar in usage to L2 regularization for XGBoost. The *spatial dropout* layer, in the text channel, drops a vector of features (ie: sentences), whereas the dropout layer, in the numeric channel, drops individual observations. The spatial dropout is preferred for the text data as we would not want to drop out individual words from a sentence (Tompson et al., 2014).

In addition, both channels include *batch normalization* layers are also included in both channels to improve training speed and stability (Loffe & Szegedy, 2015). This layer normalizes the feature space during *batch* training, a technique used to iteratively train deep learning models on a subset of data until the model has seen the entire training set; a full cycle through the training set is called an *epoch*.[42]

J. Additional NLP results

Table J1: Logit Classification Report

	Negative	Neutral	Positive	Accuracy	Macro average	Weighted average	AUC
Precision	0.710	0.776	0.676	-	0.721	0.748	-
Recall	0.493	0.907	0.450	-	0.616	0.757	-
F1-score	0.582	0.836	0.540	-	0.653	0.742	-
Support	893	3408	894	0.757	5195	5195	0.854

Notes: Negative, Neutral, and Positive are the sentiment categories. Each row shows the individual measure (e.g., Precision) for each category. Macro average considers all three classes but is unweighted. Weighted average accounts for imbalance. Accuracy is not adjusted for target imbalance. Support displays the number of observations by category. All results are on the test sample.

Table J2: Random Forest Classification Report

	Negative	Neutral	Positive	Accuracy	Macro average	Weighted average	AUC
Precision	0.817	0.757	0.736	-	0.770	0.764	-
Recall	0.390	0.952	0.398	-	0.580	0.760	-
F1-score	0.528	0.844	0.517	-	0.629	0.733	-
Support	893	3408	894	0.760	5195	5195	0.866

Notes: Refer to table J1.

[42] Due to the resources required to train deep model, batch training is a commonly used method. With batches the model learns to update its weights and biases (ie: coefficients and intercepts) from a pass on a smaller batch set, rather than the full training set. The model repeats this process until it has covered the full training set, through training on batches. Each passthrough of the entire training sample is called an epoch. Multiple epochs are used during training.

K. Additional Forecasting results

Table K1: Prime results

	(1)	(2)	(3)
R-squared	0.1125	0.4711	0.6192
R-squared Adj.	0.1028	0.4593	0.5926
const	-0.0063	-0.0075	-0.1798***
	(0.0429)	(0.0326)	(0.0524)
Sentiment	1.8154**	1.2614**	1.0473**
	(0.8127)	(0.5604)	(0.5124)
Prime rate (t-1)		0.6072***	0.6176***
		(0.1262)	(0.0847)
CPI			0.1763***
			(0.0469)
Exchange Rate			0.0039
			(0.0076)
Real GDP			0.1234**
			(0.0541)
Unemployment			0.3860***
			(0.1262)

Standard errors in parentheses.

* p<.1, ** p<.05, ***p<.01

Notes: Column 1 includes only the sentiment indicator. Column 2 adds the lagged dependent. Column 3 includes all other macroeconomic controls. Column 4 removes the lagged dependent as a control.

Table K2: Mortgage 1-year results

	(1)	(2)	(3)
R-squared	0.1265	0.3831	0.5124
R-squared Adj.	0.117	0.3694	0.4784
const	-0.019	-0.0145	-0.1747***
	(0.0385)	(0.0319)	(0.0481)
Sentiment	1.7589**	1.3883**	1.1044**
	(0.7447)	(0.5526)	(0.5392)
Mortgage (1-year) (t-1)		0.5122***	0.4233***
		(0.1433)	(0.1155)
CPI			0.1650***
			(0.0577)
Exchange Rate			0.0061
			(0.0078)
Real GDP			0.1084**
			(0.0458)
Unemployment			0.2714***
			(0.1046)

Standard errors in parentheses.

* p<.1, ** p<.05, ***p<.01

Notes: Refer table K1. The regressive process for 1-year mortgages remains strong across all specifications.

Table K3: Mortgage 3-year results

	(1)	(2)	(3)
R-squared	0.1519	0.2258	0.424
R-squared Adj.	0.1427	0.2086	0.3839
const	-0.029	-0.0245	-0.1874***
	(0.0348)	(0.0341)	(0.054)
Sentiment	1.7736**	1.6268**	1.2886***
	(0.6914)	(0.6704)	(0.4912)
Mortgage (3-year) (t-1)		0.2771*	0.1617
		(0.1482)	(0.1176)
CPI			0.1911***
			(0.0502)
Exchange rate			0.0104
			(0.0089)
Real GDP			0.0743
			(0.0495)
Unemployment			0.1885*
			(0.1112)

Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
Notes: Refer table K1. The regressive process for 3-year mortgages is weak across all specifications.

Table K4: Mortgage 5-year results

	(1)	(2)	(3)
R-squared	0.1135	0.1271	0.3458
R-squared Adj.	0.1039	0.1077	0.3002
const	-0.0247	-0.0244	-0.1528***
	(0.0271)	(0.028)	(0.0422)
Sentiment	1.1893**	1.1252**	0.7276*
	(0.532)	(0.5459)	(0.393)
Mortgage (5-year) (t-1)		0.1292	0.0525
		(0.1296)	(0.1083)
CPI			0.1365***
			(0.0398)
Exchange rate			0.0160**
			(0.0081)
Real GDP			0.0769**
			(0.0356)
Unemployment			0.1897**
			(0.0797)

Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
Notes: Refer table K1. The regressive process for 5-year mortgages is weak across all specifications.

Table K5: CORRA with lagged Sentiment (t-1)

	(1)	(2)	(3)
R-squared	0.0581	0.4291	0.6044

R-squared Adj.	0.0478	0.4164	0.5768
const	-0.0118	-0.0033	-0.1955***
	(0.0435)	(0.0334)	(0.052)
Sentiment (t-1)	1.3289**	0.1212	-0.0782
	(0.662)	(0.4652)	(0.3946)
CORRA (t-1)		0.6462***	0.6347***
		(0.1185)	(0.0745)
CPI			0.1899***
			(0.0449)
Exchange rate			0.0117
			(0.0088)
Real GDP			0.1434***
			(0.0552)
Unemployment			0.4288***
			(0.1267)

Standard errors in parentheses.

* p<.1, ** p<.05, ***p<.01

Notes: Refer to table 7. Here our sentiment metric is explicitly lagged one-period. This results in an insignificant result in columns 2 and 3.

Table K6: All results without the sentiment metric

	CORRA	Prime rate	Mortgage 1-year	Mortgage 3-year	Mortgage 5-year
R-squared	0.6042	0.5871	0.4696	0.3554	0.3091
R-squared Adj.	0.5815	0.5634	0.4391	0.3183	0.2694
const	-0.1960***	-0.1892***	-0.1836***	-0.1979***	-0.1593***
	(0.0525)	(0.0547)	(0.0483)	(0.0602)	(0.0448)
CORRA (t-1)	0.6318***				
	(0.0729)				
Prime rate (t-1)		0.6340***			
		(0.0761)			
Mortgage (1-year) (t-1)			0.4424***		
			(0.1052)		
Mortgage (3-year) (t-1)				0.1849	
				(0.1151)	
Mortgage (5-year) (t-1)					0.0637
					(0.1069)
CPI	0.1894***	0.1790***	0.1668***	0.1934***	0.1388***
	(0.0443)	(0.0437)	(0.0535)	(0.0502)	(0.0384)
Exchange rate	0.0111	0.0106	0.0132	0.0188*	0.0207**
	(0.0083)	(0.008)	(0.0082)	(0.0102)	(0.0085)
Real GDP	0.1443**	0.1472**	0.1333***	0.1037*	0.0935**
	(0.0565)	(0.0589)	(0.0491)	(0.0555)	(0.0377)
Unemployment	0.4314***	0.4314***	0.3167***	0.2416**	0.2197***
	(0.1298)	-0.1353)	(0.1103)	(0.1205)	(0.0816)

Standard errors in parentheses.

* p<.1, ** p<.05, ***p<.01

Notes: Each column shows the results for a given interest rate series; the lagged dependent is included in all specifications. Sentiment is strictly excluded to show the fit with standard macro-indicators alone. The lower R-squared Adj. – across all columns – show that our standard indicators are not able to explain as much of the variation in interest rate changes than with the inclusion of sentiment. This result strongly suggest that sentiment is offering additional information useful for forecasting policy changes.