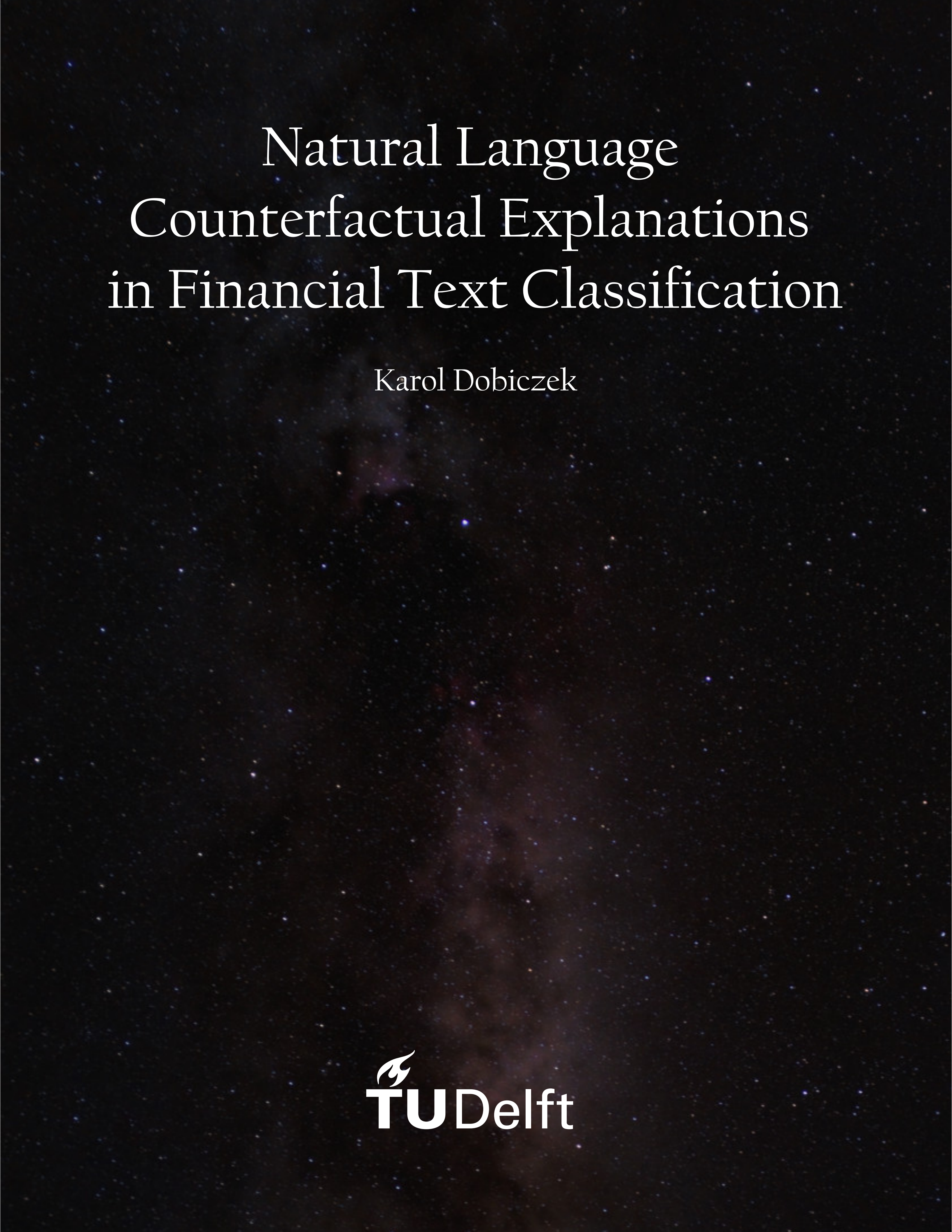


Natural Language Counterfactual Explanations in Financial Text Classification

Karol Dobiczek

Natural Language Counterfactual Explanations in Financial Text Classification

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Karol Dobiczek
born in Katowice, Poland

Thesis Committee:	Dr. Cynthia C. S. Liem MMus	TU Delft, daily supervisor
	Patrick Altmeyer MSc	TU Delft, daily co-supervisor
	Dr. Jie Yang	TU Delft

Intelligent Systems
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Preface

This thesis is the culmination of over nine months of work. Over this time, it has gone through many stages and evolved, taking directions that I would have never foreseen. Although challenging, it has also been a journey of discovery and a great learning experience. I owe it to those around me, and I would like to thank all of you.

First, I would like to thank my supervisors, Dr. Cynthia Liem and Patrick Altmeyer. With your guidance, feedback, and a friendly and positive attitude, working on this project was a pleasure.

Cynthia, thank you for letting me dive into a topic I was unfamiliar with and experiment with it. I am really grateful for your guidance, helping me stay on the right track even when I tried going into the deepest of rabbit holes and for giving me hope and support along the way. Your comments helped me change my point of view on the project and improve it by incorporating many aspects that I would not have noticed otherwise.

Patrick, thank you for the time and effort you put into countless meetings and for the encouragement that helped me complete this thesis. Thank you for helping me find this topic and for helping me follow through with it. I am very grateful for your help in recruiting the expert survey participants, whose opinions turned out to be a crucial part of the thesis. I admire the dedication, care, and contagious enthusiasm with which you approach your projects.

I want to thank the people who contributed to this work. Andrew, for his help in theorizing and deploying user studies, Alex, for saving me countless hours of compiling the surveys, and Richard, for his patience and help in completing the HREC.

I also want to give special thanks to the many central bank employees, including the Federal Reserve Board staff, who devoted their time to helping assess text counterfactual explanations.

I want to give thanks to all my friends and family for their support throughout this journey.

To Aleksander, thank you for sticking around throughout the five years of our studies in Delft. I am very grateful for the countless hours of conversations, brainstorming, our lunches, bouldering sessions, and all the feedback and support I have received from you during those years.

To my friends Han, Hubert, Kuba, Olek, Jacek, and Ksawery, whom I have known for years, thank you for all the fun times we had together.

To Ola, Natalia, Michał, Filip, Filip, Kuba, and Pieter for keeping me company during my time in the Netherlands.

To Fatemeh, Yaren, Oscar, Kristóf, Fernando, Alex, and Francesco for watering my plants and helping me always feel at home in Delft.

To my lovely girlfriend Zosia, thank you for your constant support and love, for bearing with me during my work on this thesis, and for making me take a break from time to time.

Finally, I want to thank my brother Bartek and my parents, Ewa and Tomasz, for their support, patience, and care.

Karol Dobiczek

Delft, the Netherlands
30th August 2024

Abstract

Central banks communicate their monetary policy plans to the public through meeting minutes or transcripts. These communications can have immense effects on markets and are often the subjects of studies in the financial literature. The recent advancements in Natural Language Processing have prompted researchers to analyze these communications using Transformer-based Large Language Model (LLM) classifiers. The use of LLMs in finance and other high-stakes domains calls for a high level of trustworthiness and explainability of those models. We focus on Counterfactual Explanations, a form of Explainable AI that explains a model's classification by proposing an alternative to the original input. We use three types of CE generators for LLM classifiers on a recent dataset consisting of sentences taken from FOMC communications to assess the usability of their explanations. We perform three experiments comparing different types of generators, one using a selection of quantitative metrics and two involving human evaluators, including central bank employees. Our findings suggest that non-expert and expert evaluators prefer counterfactual methods that apply minimal changes to the texts; however, the methods we analyze might not handle the domain-specific vocabulary well enough to generate plausible explanations for our task. We discuss shortcomings in the choice of evaluation metrics in the literature on text CE generators and propose refined definitions of the fluency and plausibility qualitative metrics.

Contents

1 Introduction 3

1.1 Motivation 4

1.2 Thesis objective and research questions 5

1.3 Contributions 5

1.4 Thesis structure 6

2 Neural Text Classification in Finance 7

2.1 Machine Learning 7

2.1.1 Linear Models 7

2.1.2 Neural Networks 9

2.1.3 Deep Learning 12

2.2 Deep Learning in Natural Language Processing 12

2.2.1 Transformers 12

2.2.2 NLP Transformer Models 14

2.3 Text Classification in Finance 16

3 Generating and Evaluating Counterfactual Explanations for Language Models 18

3.1 Explainable Artificial Intelligence 18

3.2 Counterfactual Explanations 19

3.3 Text Counterfactual Explanations 20

3.3.1 LLM-Assisted Generation 21

3.3.2 Latent Perturbation and Decoding 22

3.3.3 Sequential Generation 24

3.4 Evaluating Text Counterfactuals 25

3.4.1 Quantitative Metrics for Text CEs 25

3.4.2 Quantitative Metrics Implementation 26

3.4.3 Qualitative Metrics for Text CEs 28

4 Experimental Setup and Evaluation Methodology 30

4.1 Task Description 30

4.1.1 Model Selection 30

4.1.2 Model Parameters 31

4.2 Experiment 1: Quantitative Metrics . . . 33
4.3 Experiment 2: Fluency of the Explanations . . . 33
4.3.1 Defining Fluency . . . 34
4.3.2 Survey Design . . . 34
4.3.3 Participant Recruitment and Sample Sizes . . . 35
4.4 Experiment 3: Expert Evaluations . . . 36
4.4.1 Defining Plausibility . . . 36
4.4.2 Survey Design . . . 37
4.4.3 Participant Recruitment and Sample Sizes . . . 37

5 Experimental Results and Discussion 39
5.1 Survey Duration . . . 39
5.2 Experiment Results . . . 40
5.2.1 Agreeance Between the Results . . . 41
5.3 Expert Insights on Counterfactuals . . . 43
5.4 Data Complexity . . . 45

6 Conclusions and Future Work 47
6.1 Conclusions . . . 47
6.2 Recommendations for Practitioners . . . 49
6.3 Future Work . . . 49
6.4 Limitations . . . 50

Appendices 60

A Informed Consent Form 61

B Survey Topic Introduction 62

C Additional Quantitative Results 64
C.1 Alternative Models for Perplexity Calculation . . . 64
C.2 Quantitative Results of Successful Counterfactuals . . . 64

D Pseudo-RAG Generator 65
D.1 Results . . . 66

Chapter 1

Introduction

> Central banks choose their words very carefully. And rightly so – policymakers' wording can move markets and, eventually, the economy.
>
> Gebauer, McGregor, and Schumacher [26]

Monetary policies of central banks aim to influence the markets to ensure stability and growth. Central banks tend to communicate their monetary policies by releasing meeting minutes or transcriptions to control factors like the inflation expectation that, in turn, influence the market's growth [57]. It is thus beneficial for stakeholders to analyze those communications and adequately react to incoming market changes. Recent developments in Natural Language Processing (NLP) gained massive popularity with advancements and applications such as text generation or sentiment analysis. Those developments have also been applied in the financial domain in analyzing and predicting future market movements [21]. The reliance on those NLP Language Models (LMs) in numerous critical domains raises the questions: *How can we trust those models?* and *How do those models interpret the data?* Explainable AI (XAI) methods try to shed light on how machine learning models interpret data, and while those methods exist for LMs, there has been little work done in applying them to expert domain fields such as finance.

Since the early 1990s, many central banks have started to disclose their strategies openly [57] through press releases, meeting minutes, transcripts, and conferences. Central banks such as the Federal Reserve Bank (Fed), the European Central Bank (ECB), or the Bank of England (BoE) aim to inform the media, governments, financial market participants, and the general public about their upcoming policies. The significance of these communications can be highlighted by how a market responds to them – reactions (such as changes in overnight index swaps) are often registered within minutes after press releases or conferences are published [1]. Communicating policies is difficult, so these communications have become more verbose [15]. Moreover, they frequently contain the central bank's assessment of the economic outlook, possibly increasing the ambiguity of the communication [32]. The complexity and format of these communications have prompted researchers to use Natural Language Processing techniques to analyze and understand central bank communications.

The NLP field has grown considerably over the past years, mainly thanks to the advancements in Artificial Intelligence (AI) model architectures and the abundance of training data. As these models reach great accuracy on many tasks, the field has shifted toward research on AI. Large Language Models (LLMs) using the Transformer architecture, such as the GPT model or Mistral, show impressive few-shot or zero-shot results [54], performing complex tasks with seemingly no retraining or fine-tuning needed. These models are particularly useful due to the data type they work on. Textual data is relatively unstructured, and language models can help users avoid a considerable amount of work. However, LLMs are often close-sourced, act as black boxes, and are thus hard to explain. Furthermore, using them in domains dealing with personal data or relying heavily on taking high stakes in their outputs requires organizations and practitioners to follow safety considerations to maintain safe use.

As the popularity of LLMs permeates the mainstream, these models start to be used in various applications, often including those facing end-users directly. Chat-based models such as Chat-GPT work well as customer service chatbots for retail websites. LLMs combined with vector text databases can generate text backed by ground-truth texts in a methodology called Retrieval-Augmented Generation (RAG). Some applications involve compiling large amounts of texts, such as legal texts or invoices, for which a company can fine-tune a pre-trained LLM to generate the required text types.

1.1 Motivation

Counterfactual Explanations (CE), a popular technique in explainable AI, explain a model's behavior by proposing an alternative to the original input that changes the model's output. If we change the model's input by, for example, changing a word in the case of text such that the model classifies this new input into a different class from the original one, we have created a counterfactual. Such counterfactuals can serve as explanations for the underlying model since they reveal to us what input changes the classifier is sensitive to.

The counterfactual approach to ML model explanations adheres well to the field of NLP, as natural language texts are easy for human readers to interpret. Therefore, a well-formed counterfactual might offer valuable insight into the model's behavior. Over the years, multiple approaches for generating counterfactual texts have been formulated: some approaches with the direct goal of generating counterfactuals that explain particular classifier models [7, 56], others using classifier models to control text generation [16] or enable data augmentation [71], that then get used to generate explanations [43].

Most of the methods in the literature of language model counterfactual explanations [1] use

[1] In this work we use the terms LM counterfactual explanations and text counterfactual explanations interchangeably.

simple datasets and tasks in training and evaluation [71]. These simple tasks often include negation or rewording. While counterfactual examples generated for those tasks might prove helpful in data augmentation, current works lack evaluation of text explanations from specialist domains. Similarly, adapting a counterfactual generator to a complex task might fail if the task we are adapting to falls out of the distribution of the usual training sets.

1.2 Thesis objective and research questions

The main goal of the thesis is to thoroughly test LM counterfactual generators' ability to generate explanations for texts in expert domains. To do this, we apply methods of generating counterfactual explanations for language models on a financial text classification dataset.

We aim to answer the following research questions:

- *RQ1: What are the essential desiderata of text counterfactual explanations for language models?*
- *RQ2: What type of explanations is most useful for the practitioners?*
- *RQ3: Are the metrics commonly used in Natural Language Processing right for evaluating and comparing text counterfactual generation methods?*

Initially, we focus on researching the trends in text counterfactual generator design in order to answer RQ 1. We study the considerations such as desiderata or qualities of text explanations. We derive three main classes of counterfactual generators based on how they generate text and study one generator from each class in our experiments.

Determining how to generate text counterfactual explanations for texts in expert domains, particularly in finance, is the main focus of our research (RQ 2). We ask fluent English speakers and central bank employees, including employees of the United States Federal Reserve Board [2] to grade the counterfactuals generated using different methods. Following their gradings, we aim to determine whether these counterfactual generators are useful and find the methods that give the most promising results.

Finally, we focus on how we organize the text counterfactual explanation methods and metrics for measuring their outputs. Using the results of our experiments, we determine which metrics are useful for text CE methods evaluation and which are not fit for comparing them (RQ 3).

1.3 Contributions

We apply counterfactual explanation generators for LM classifiers on a real-world dataset and evaluate their utility. Our evaluations involve human annotators judging the grammatical correctness of the texts generated on a novel task. Additionally, we ask central bank

[2] The research responses, analysis, and conclusions set forth are those of the authors and do not indicate concurrence by other members of the research staff or the Federal Reserve Board of Governors.

employees for their insights on whether the changes introduced by the counterfactuals are realistic for the domain. Our findings suggest future research directions in counterfactual generation for text classification in specialist fields.

The literature on LM counterfactuals employs a variety of metrics used in NLP. We collect a number of those metrics, both quantitative and qualitative, and reason about their usability in evaluating text counterfactuals. We propose a new definition of two qualitative metrics: fluency and plausibility. Fluency refers to the general correctness of a sentence, while plausibility measures whether it is realistic in the context of the domain and the expected sentiment class. We establish concrete definitions of these metrics. With those, we aim to remove some of the guessing work that might occur for human annotators.

We find that most of the quantitative metrics used in assessing LM counterfactuals do not correlate with the fluency or plausibility of the explanations. However, we find a strong dependence between the fluency of an explanation and the edit distance metrics, suggesting that minimal changes are preferred. Through comments of the central bank employees, we find that using domain-specific wording incorrectly can make a counterfactual less plausible than not using this type of wording at all.

1.4 Thesis structure

We divide this thesis into six chapters. Chapter 2 discusses the use of language models for financial text classification. In Chapter 3, we introduce explainability in machine learning and describe the methods used in this thesis. Our experimental setup is described in Chapter 4, and the results of our experiments are discussed in Chapter 5. Finally, we conclude the thesis with Chapter 6, summarizing our findings, stating the limitations of our work, and the possible future avenues of research in this field.

Chapter 2

Neural Text Classification in Finance

This section gives an overview of the methods and background used in our research. Section 2.1 gives an overview of the background of machine learning. In subsection 2.2, we introduce modern ML models used in Natural Language Processing. Finally in Section 2.3, we discuss how machine learning and NLP are applied in financial text classification.

2.1 Machine Learning

Machine learning is a field of computer science and statistics concerned with creating algorithms called models that automatically adapt to the data they are applied to perform given tasks. Machine learning can be split into three main types: supervised, unsupervised, and reinforcement learning. Supervised learning algorithms adapt data with set input and output features. In unsupervised learning, the algorithm is tasked with finding patterns in data with no previously defined output values, such as finding clusters in data. Reinforcement learning consists of algorithms that represent agents in a simulated environment and learn optimal sets of actions. In this thesis, we will introduce and focus on supervised learning algorithms. The equations in this section are based on Bishop [11].

2.1.1 Linear Models

A common task for a supervised learning algorithm is regression. Given training data consisting of observations $x = (x_1, ..., x_k)$ and their corresponding target values $y = (y_1, ..., y_k)$, a regression model should give a prediction for a new, unseen value of x. A regression model represents a function $f(x)$ that maps observations x to their output values y. From a probabilistic point of view, a model can represent a conditional probability distribution $p(y|x)$.

The simplest model for the regression task is a linear model which outputs a linear combination of the input values with a vector of weights $w = (w_0, ..., w_k)$:

$$f(x) = w_0 + w_1 x_1 + ... + w_k x_k = \sum_i^k w_i x_i$$

The formula can be simplified to $f(x) = x^T w$ if a 1 is appended to the beginning of x, $x = (1, x_1, ..., x_k)$. The output of the function is the prediction of our linear model. If our task involved binary classification, i.e., $y \in \{0, 1\}$, the output of the function needs to be transformed using a logistic sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

After applying the sigmoid function to a linear model, we get logistic regression. In the probabilistic view, linear regression models the $p(y|x)$ as a Gaussian distribution, while logistic regression models the conditional as a Bernoulli distribution [49].

Training a Linear Model

The weights need to be fitted to the training data to get predictions adapted to it. To this end, we define an error measure that establishes the rules that penalize errors between the model's predictions and the actual target values. This measure is often called an error function or a loss function and quantifies the quality of a fit of the model to the training or test data. A common choice for this function is a square loss, where the difference between the prediction and the true target value is squared. A loss function can then be defined as follows:

$$\mathcal{L} = \sum_i^n (\mathbf{x}_i^T w - \mathbf{y}_i)^2 = ||X^T w - Y||^2 \tag{2.1}$$

with $\mathbf{x}$ being a set of n observations and $\mathbf{y}$ being their corresponding output values. The resulting loss function can then be solved to achieve optimal weight fit. First, we take a derivative with respect to w to achieve the gradient of the loss function:

$$\nabla_w \mathcal{L} = 2X^T X w - 2X^T Y$$

then we find the w for which the gradient is equal to 0:

$$\begin{aligned} 2X^T X w - 2X^T Y &= 0 \\ w &= (X^T X)^{-1} X^T Y \end{aligned}$$

This analytical solution is possible due to the rather simple definition of the model. It is not always possible to arrive at an analytical solution, and training more complicated models is often carried out via gradient descent (Subsection 2.1.2).

This linear model can be useful for solving a regression problem for simple, linear relations between the input variables. An improvement we can make is to transform the input data using a basis function $\phi(x)$. A non-linear transformation can help the model learn convex functions, which can help with some tasks. However, more complex tasks, like natural text classification, require the model to learn representations of its inputs that will allow for successfully performing its task. A model architecture widely used and adopted as a general solution to many problems is the multilayer perceptron or neural network architecture.

2.1.2 Neural Networks

A neural network model consists of a series of connected layers. Connecting a series of simpler models allows the model to transform the data at each layer, possibly extracting information that will make it easier to complete the tasks for later layers. This architecture is known as a feed-forward network.

The first layer in a feed-forward network accepting inputs $x = (x_1, ..., x_D)$ can be formalized as a series of segments:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i, \quad z_j^{(1)} = h(a_j^{(1)}) \tag{2.2}$$

with $j = 1, ..., M$, M being the number of outputs, and (1) signifying the first layer of the network. The coefficients w_{ji} are called weights, and the terms w_{j0} are called biases. The linear function outputs a_j are called activations. The activations are then fed through a differentiable activation function $h(\cdot)$ that introduces a non-linearity to the layer's outputs and produces the hidden representation z_j. The hidden representations of a layer are used as inputs to the next one:

$$a_j^{(2)} = \sum_{i=1}^{K} w_{ji}^{(2)} a_i^{(1)}, \quad z_j^{(2)} = h(a_j^{(2)})$$

for K representations, $j = 1, ..., K$. For the last layer of the network, i.e., the one whose outputs should match the target values y, the activation function is not applied. Instead, the pre-activation value of the layer's output can be used for regression (directly) or binary classification (after sigmoid transformation).

If our task involves more than two classes, the last layer should have a number of outputs matching the number of classes with a class label assigned to each output in training. To transform the class activations to predicted class probabilities, we apply a smoothing softmax function:

$$y_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

for j output activations, where y_k is the kth class probability. The predicted class is the one assigned to the output, achieving the softmax value.

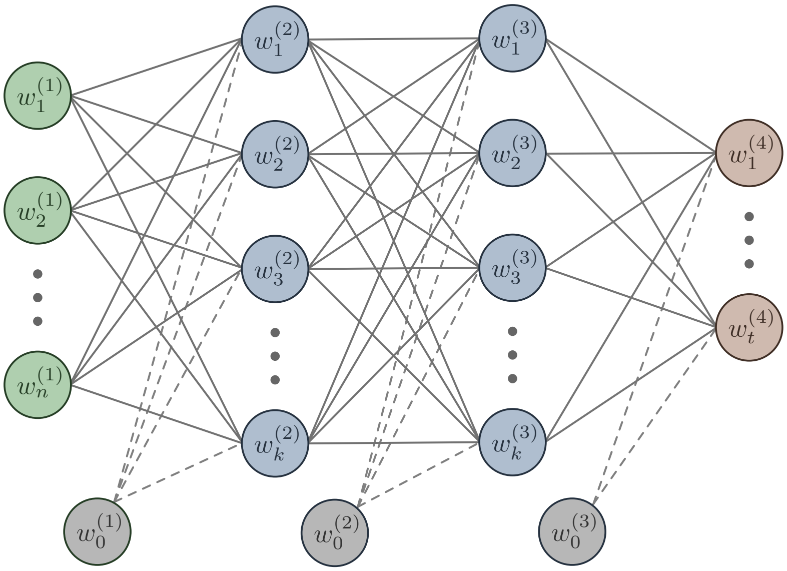


Figure 2.1: A diagram of a four-layer neural network. The green nodes represent inputs, the red nodes represent outputs, the blue nodes are the two hidden layers, and the grey nodes represent biases.

A neural network composed of two hidden layers of size k with input size n and output size t, like the one pictured in Figure 2.1, results in the following function:

$$y_k = \sum_t^T w_t^{(4)} \ h\left(\sum_k^K w_k^{(3)} \ h\left(\sum_k^K w_k^{(2)} \ h\left(\sum_n^N w_n^{(1)} x_n\right)\right)\right) \tag{2.3}$$

Training a Neural Network

Similar to the linear model training procedure introduced in Subsection 2.1.1, to train the neural network, we first need to define a loss function and optimize the neural network's weights to minimize the loss value. We can assume, ex., a square error loss (Equation 2.1) and calculate its gradient $\nabla\mathcal{L}$. However, due to the complexity of our model, solving for $\nabla\mathcal{L} = 0$ is infeasible. Additionally, this gradient might be close to or at 0 in many for many values of weights. Considering the space of possible values the weight vector can take, these points represent local minima. Ideally, we want to avoid those and find the global minimum with the lowest loss function value.

The gradient descent optimization procedure is an iterative approach to loss function optimization. At each iteration, the gradient of the loss function is computed and applied to the weights, taking a step towards the loss-minimizing direction. Given model weights w^t at a timestep t, the new weights are computed as:

$$w^{t+1} = w^t - \eta \nabla \mathcal{L}(w^t)$$

where $\mathcal{L}(w^k)$ is the loss function evaluated for a model with weights w^k and η is a scaling factor called the step size.

Backpropagation

The final step in the training procedure is the calculation of the loss gradient. In neural networks, this calculation is done using a procedure called backpropagation. The forward pass (inference) passes the input through the weights of the network to produce the output. Since the layers are fully connected, the output of a single activation affects all the inputs of the next layer. Starting from the last layer, Backpropagation calculates the gradient w.r.t. a weight by combining the information about the gradients of the next layer's weights, propagating the gradients backward.

The last layer outputs the model's predictions. Thus, the gradient calculation will be dependent on the loss function. Considering outputs y_k, targets $\hat{y}_k$ and the square loss function

$$y_k = \sum_i w_{ki} x_i, \quad \mathcal{L} = \frac{1}{2} \sum_k \left(y_{nk} - \hat{y}_{nk}\right)^2$$

we can calculate the gradient w.r.t. a weight w_{ji} as

$$\frac{\partial \mathcal{L}}{\partial w_{ji}} = (y_{nj} - \hat{y}_{nj}) x_{ni}$$

which constitutes the error signal that will be propagated to the rest of the weights. For the rest of the layers, the backpropagation algorithm takes advantage of the chain rule for derivatives:

$$\frac{\partial \mathcal{L}}{\partial w_{ji}} = \frac{\partial \mathcal{L}}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}, \quad \delta_j \equiv \frac{\partial a_j}{\partial w_{ji}}$$

and since the outputs z_j of hidden layers are simply a weighted sum of the inputs (Equation 2.2), then we can calculate $\frac{\partial \mathcal{L}}{\partial a_j} = z_i$, so:

$$\frac{\partial \mathcal{L}}{\partial w_{ji}} = \delta_j z_i$$

Finally, for a hidden layer, we take into account the backpropagated gradients:

$$\delta_j = \sum_k \frac{\partial \mathcal{L}}{\partial a_k} \frac{\partial a_k}{\partial a_j} = \sum_k z_j w_{kj} \delta_k$$

2.1.3 Deep Learning

With efficient methods for training neural networks, researchers have started creating deeper networks containing more hidden layers. Deep learning refers to those models. There are numerous deep-learning architectures

Convolutional neural networks (CNN) learn and apply convolutions or filters that extract features from images. Using fewer weights and connections, those networks can efficiently process multidimensional inputs and detect spatial dependencies in the data.

Recurrent neural networks (RNN) are designed to process sequential data such as texts, time series, or speech. They differ from the usual feed-forward networks by processing the input sequence one part at a time and maintaining a single hidden state. By updating the hidden state based on the incoming inputs and the previous hidden states, the RNN can essentially memorize (it is autoregressive).

A new addition to the family of deep learning models is the Transformer architecture described in Section 2.2.1. The model solves many of the issues related to using previous architectures, such as the RNN, and allows for efficient pre-training on large data corpora. These pre-trained models often serve as foundations for further training or fine-tuning toward their use in specific tasks.

2.2 Deep Learning in Natural Language Processing

Natural Language Processing deals with a wide array of tasks related to processing and understanding natural language, from the most basic ones like breaking texts into tokens, parsing, and analyzing the structure of a sentence to understanding the meaning of words and finding relationships between words and sentences.

As NLP tools, we can consider packages like NLTK [10] performing tokenization tasks, part-of-speech tagging, parsing words into syntax trees based on mathematical formulations, and many more. We can also consider ML models such as word2vec [47] that try to learn semantic vector representations of tokens based on their typical relations with surrounding words.

2.2.1 Transformers

The Transformer architecture [66], widely used in our work, has sparked a major shift in the NLP models architecture and has allowed for many advantages leading up to the modern LLM. One of the key changes introduced in the transformer architecture is the transition from recurrent and convolutional layer-based architectures toward attention (Subsection 2.2.1). Recurrent neural network (RNN) models encode an input sequence by calculating a hidden embedding h_t for each position t in the sequence by combining it with the previous embedding h_{t-1}. This means an embedding h_t depends on the last hidden state. Consequently, parallelizing these models' training and inference processes is impossible, which is a major drawback, especially for long input sequences. Unlike most successful architectures before the Transformer, it uses only the attention block with some modifications.

Self-Attention

Initially, the attention method was introduced for neural machine translation [6] to enable a neural network to learn relations between sequences of words over long input spans.

The main idea behind attention is to let the model learn how to relate input words to other words through three weight matrices: query matrix W_Q, key matrix W_K, and value matrix W_V. Given an input sequence X, the attention mechanism relates a vector representation of a single token $\underline{x}$ of dimension d_k to the rest of the input sequence by first calculating the $Q = \underline{x}W_Q$ query, $K = XW_K$ key and $V = XW_V$ value matrices. The product of Q and K^T gives a vector of "soft weights" α for the V matrix. $Attention(Q, K, V) = QK^TV$ then gives the values of the input tokens in X correlated with $\underline{x}$. Vaswani et al. [66] formulate a particular version of self-attention called the "Scaled Dot-Product Attention" pictured by Figure 2.2a, wherein the soft weights are scaled by first dividing them by $\sqrt{d_k}$ and then applying the Softmax function on the result. The resulting value weights can then be applied to the V matrix forming:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.4}$$

The rationale behind the division by $\sqrt{d_k}$ as stated by Vaswani et al. [66] is that for large values of d_k the dot products of the query and key matrices have large magnitudes causing the Softmax to be very small.

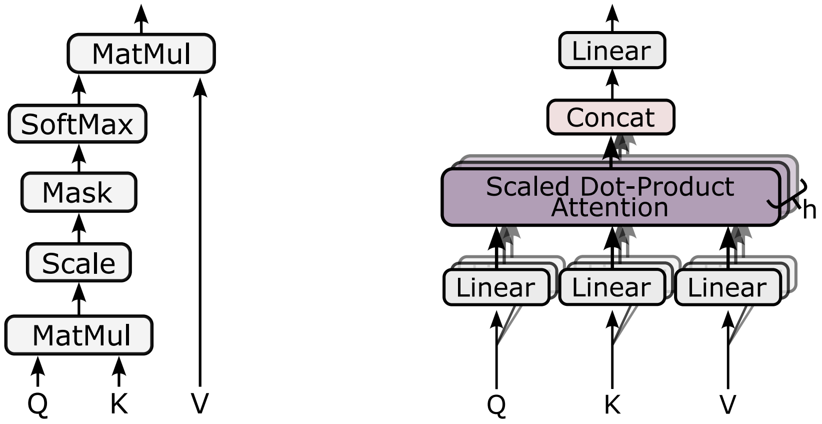
Multi-Head Attention

A single attention layer maps the input to a d_{model}-dimensional representation. Still, it is limited by only learning an averaged representation of the relations encountered in the training data. As illustrated by Figure **??**, authors of the Transformer model mitigate this issue by linearly projecting the Q, K, and V representations to d_q, d_k, and d_v dimensions, respectively, into h versions of projections called heads, each with separate learned projection. This allows the model to capture relations at different representation subspaces and positions. The authors formulate the multi-head attention as follows:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_i)W^O, \tag{2.5}$$

$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.6}$$

The projection matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_q}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, and $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W^O \in \mathbb{R}^{d_v \times d_{model}}$ mapping the output to the d_{model} dimension.



(a) Scaled Dot-Product Attention

(b) Multi-Head Attention

Figure 2.2: The Scaled Dot-Product Attention (a) and Multi-Head Attention (b) composed of several attention layers. Adapted from [66].

2.2.2 NLP Transformer Models

Since the definition of the Transformer architecture, numerous language models have been developed that use it. These models can have many main tasks, such as text translation, text-to-text generation, mask infilling, or text classification. This section discusses two of those Transformer models, BERT and GPT, used in various LM counterfactual generators described in Section 3.3.

Generative Pre-trained Transformer

The Generative Pre-trained Transformer introduced by Radford et al. [53] is a Transformer model achieving state-of-the-art performance on various tasks at its release. The authors use unsupervised pre-training, a token prediction task where given tokens $\mathcal{U} = \{u_1, ..., u_n\}$, the model optimizes the following likelihood:

$$\mathcal{L}(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, ..., u_{i-1}; \theta)$$

for a context window k, conditional probability P, and model parameters θ.

The model is also fine-tuned on the following supervised tasks: Natural Language Inference, Question Answering, Sentence Similarity, and Classification. The authors use a dataset of labeled texts $\mathcal{C}$ and optimize

$$\mathcal{L}(\mathcal{C}) = \sum_i \log P(y_i | x_i^1, ..., x_i^m)$$

where y is the predicted label.

The authors use a 12-layer decoder Transformer with 12 attention heads for each layer and 768-dimensional (d_{model} in Section 2.2.1) states for each head. This and the position-wise feed-forward networks give a parameter count of 117 million. Radford et al. [54] later released GPT-2, an architecture based on GPT, introducing a few modifications and increasing the vocabulary, context size, amount of layers, and d_{model} dimensionality. The largest variant of the GPT-2 had a parameter count of 1.5 billion and again achieved zero-shot state-of-the-art results on most tasks.

Bidirectional Encoder Representations from Transformers (BERT)

The GPT model proved that the Transformer architecture performs very well, even for zero-shot tasks. However, Devlin et al. [19] see one main shortcoming of GPT: the left-to-right architecture for language modeling, where a token can only attend to previous tokens in self-attention. They argue that the left-to-right approach is "sub-optimal for sentence-level tasks and could be very harmful when applying fine-tuning based approaches to token-level tasks ...".

Devlin et al. [19] introduce Bidirectional Encoder Representations from Transformers (BERT), an architecture that maintains bidirectionality by using a masked language model (MLM) objective. BERT is trained on the Masked LM task and the Next Sentence Prediction task.

In the Masked LM task, the authors mask 15% of the training set tokens. The model is then tasked to predict the masked tokens by predicting the token's ID, similarly to the classification task in Subsection 2.1.2. Depending on the downstream task, the masks might not appear in the fine-tuning stage, creating a mismatch. To mitigate that, out of the 15% of the training data selected for masking, 80% of the tokens are masked, 10% are filled in with a random token, and for another 10%, the original token stays unchanged.

The second task, Next Sentence Prediction, involves training the model to detect relationships between sentences. The authors prepare sets of two sentences, where 50% of the time, the second sentence follows the first in the original dataset, labeled as `IsNext`. In the other 50% of the cases, the second sentence is a random sentence from the training corpus and is labeled as `NotNext`.

Similarly to the GPT model, BERT uses 12 Transformer layers, each with 12 self-attention heads and d_{model} of 768 (110 million parameters in total) for the base variant. The large variant has 24 layers, 16 self-attention heads, and d_{model} of 1024 (340 million parameters).

In our work, we analyze a classification model called RoBERTa, or the Robustly optimized BERT approach. RoBERTa [39] is a configuration of the BERT model with a modified pre-training procedure. RoBERTa uses dynamic masking, replaces the next sentence prediction task with one that uses more sentences at once, drops the NSP loss, uses large mini-batches, uses a different training set, and increases the number of training passes. RoBERTa achieves state-of-the-art results at release time using the same architecture as BERT.

RoBERTa is adapted to classification tasks by removing the MLM layer and replacing it with a fully connected neural network. This model can then be fine-tuned for the specific classification task.

2.3 Text Classification in Finance

Machine learning plays a vital role in the financial domain. It is used in a wide array of applications, from statistical modeling in price prediction through deep learning in risk assessment to modeling and simulation in policy assessment [14]. Due to a large part of financial data being in the form of texts, natural language processing techniques have proven helpful in tasks such as NLP-based financial forecasting [73] or market prediction [36].

Depending on the task, researchers use various models in their analyses: Tetlock, Saar-Tsechansky, and Macskassy [62] aim to predict changes in stock prices by analyzing the fractions of negative words contained in articles; Bi [9] uses a Hidden Markov Model to segment texts to classify their sentiment using an LSTM; Delice et al. [18] use Latent Dirichlet Allocation along with several language models to determine the polarity of sentences and documents; and Jarociński and Karadi [32] assess the impacts of monetary policy shocks and the central bank information shocks using Bayesian structural autoregression. In this work, we focus on central bank communication analysis.

Through communications such as meeting minutes or transcripts, central banks inform the public about their upcoming policies. This approach to controlling inflation expectations and shaping interest rates has only started to be used in the 1990s [57], and its relevance has still been on the rise over the past years with the growing complexity of the policies conveyed in the communications [15]. Monetary policy communications from organizations like the BoE, ECB, or the Federal Open Market Committee (FOMC) play an important role in financial text analysis: Frunza and Stanley [24] extract concept-value pairs from FOMC communications using a ranking algorithm. [45] use word embeddings and vector autoregression to extract semantic changes in policy documents. Mathur et al. [44] aim to exploit the multimodal monetary policy calls by gathering a dataset comprised of visual, audio, and textual data and training a model to forecast the financial risk movement associated with the data.

Among the central bank communication sentiment analysis tasks, we focus on the classification of financial texts into *hawkish* or *dovish*. This classification relates to the perception of the communication – a *hawkish* perception suggests that the expectations are on the tightening side (decreasing money supply), while the *dovish* perception suggests a loosening side (increasing money supply) [63]. Numerous works have attempted to analyze central bank communications through the hawkish-dovish lens, employing simple classifiers such as the Support Vector Machine (SVM) [63], through modern Transformer-based models such as BERT [69] and RoBERTa [52]. Others turn to LLMs such as GPT-3.5, GPT-4 [59], or even chat-based ChatGPT. Hansen and Kazinnik [29] analyze the performance of the GPT-3 and GPT-4 on a number of FOMC texts, finding that they outperform other models and are even able to provide explanations for their classifications. Peskoff et al. [51] compile a set of FOMC documents from years between 1994 and 2016 and analyze different types of communications using the latest GPT models. Similarly, Shah, Paturi, and Chava [58] compile and release a dataset comprised of meeting minutes and speeches from 1996-2022, and press conferences from 2011-2022 annotated into three classes: *dovish*, *hawkish* and *neutral*. They also fine-tune a RoBERTa-large model, outperforming a GPT-3.5-Turbo. They compare the fine-tuned model's predictions to the Consumer Price Index (CPI) and Producer Price Index (PPI) inflation measures from the given dates.

The shift toward larger deep learning-based models in financial text analysis makes it harder for researchers to interpret what makes those models classify the texts as *hawkish* or *dovish*.

Chapter 3

Generating and Evaluating Counterfactual Explanations for Language Models

In this chapter, we introduce methods for generating and comparing explanations of the decision-making processes in language models. We introduce the concept of Explainable AI and its types and present example methods in Section 3.1. In Section 3.2, we describe counterfactual explanations in detail, and later, in Section 3.3, we discuss the approaches to generating them for language models. Finally, we present the various evaluation strategies for LM counterfactual explanations in Section 3.4.

3.1 Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) is a highly active field of research in AI, focusing on enhancing the understanding of AI systems [40]. With the ever-growing complexity of modern machine learning architectures such as CNNs or Transformers, it is infeasible to interpret them even knowing their parameters. This complexity makes those models operate essentially like *black boxes* – accepting inputs and outputting predictions without the user's insight into the decision-making process [68]. With the popularity of deep learning models being used in critical domains such as finance and healthcare, there is a steady drive towards exploring explainability in machine learning.

The approaches to Explainable Artificial Intelligence can be categorized in several ways [60]. One relates to the stage at which we want to establish explainability, either (1) in designing a model to be explainable or (2) using methods to explain a possibly *black-box* model after its training. Those two categories are called *ante-hoc* and *post-hoc* explainability respectively. An example of an ante-hoc explainable model is the linear regression model, whose parameters can directly relate to its simple formula (Section 2.1.1) and enable interpretation of the model's output. Complex models usually require post-hoc explanations, for example, employing counterfactual explanations (Section 3.2). Another way to divide explanations is whether the reason for a single prediction of a model is explained (*local explanations*) or the whole model is explained as a whole (*global explanations*).

In this work, we explore post-hoc local explanations for language models in the form of counterfactual explanations, which we describe in detail in Section 3.2. Other notable post-hoc local explanation methods include feature attributions, such as Integrated Gradients [61].

3.2 Counterfactual Explanations

Counterfactual explanations are one of the most popular forms of explaining predictions made by an ML algorithm. They aim to explain the decision made by an ML system by proposing an alternative scenario (set of inputs) that would lead the model to produce a different output. A widely used example for a CE introduced in [68] refers to a scenario where a bank refuses to provide a loan to an applicant following a decision made by an ML risk assessment system. The bank can explain the decision by giving the following CE: "You were denied a loan because your annual income was £30,000. If your income had been £45,000, you would have been offered a loan." The example shows a factual sentence and a contrasting counterfactual one that describes the feature that needs to change for the outcome to be different. Although one could argue that this example might not be entirely realistic, it illustrates the concept of CEs well.

Over the years, numerous algorithms providing automatic generation of CEs for specific ML models have emerged. Those algorithms, called counterfactual generators, employ various principles and desiderata in generating counterfactual examples. One of the most basic approaches to CE generation, formulated by Wachter, Mittelstadt, and Russell [68], states that a counterfactual should consist of a minimal perturbation of the original sample that changes the output classification. The approach can be described with the following equation:

$$\arg\min_{x'} l(x', y') + \lambda d(x, x') \tag{3.1}$$

where l is a loss function of a sample w.r.t. a target class, d is a distance function, and λ a penalty term.

This counterfactual generator often serves as a baseline or starting point for various counterfactual generation methods employing various desiderata. Examples of those methods include DICE [48], which aims to generate *diverse* counterfactual explanations with the hopes that one of them is usable to the user. CLUE [4] and REVISE [34], which train a variational autoencoder (VAE) to learn the generative process of the data, then perform a search in the latent space to decode the resulting counterfactual. Those generators tackle two different desiderata: CLUE optimizes for low predictive uncertainty of a probabilistic classifier, while REVISE aims for plausible counterfactuals.

The use of counterfactual generators has also resulted in debates on the consequences or costs of employing counterfactuals and whether the recourse taker or the recourse giver should take the costs [2]. Other approaches measure the reliability of counterfactual explanations over time [23].

Counterfactual generators have also been applied to other data modalities, such as images and text. However, due to the highly structured characteristics of these domains, a simple approach, like the one presented in equation 3.1 might produce noisy results that do not fall into the general desiderata seen in most counterfactual methods. Noisy counterfactual examples resembling adversarial attacks rather than actual samples from the original data distribution, thus having low plausibility [3], might not serve as usable explanations of the model. Hence, developing generators specialized for more complicated data modalities or even for select models is necessary. The task of generating counterfactual explanations for unstructured data becomes more generative than usual.

The plausibility of counterfactual explanations, discussed in Subsection 3.4.3, is the closeness of the explanation to the original data distribution. Plausibility in CEs is crucial in applications such as algorithmic recourse, where an explanation acts as a means for an individual negatively affected by an AI system's prediction to change their outcome by employing the explanation. However, if an explanation is highly implausible, it might be impossible to follow, for example, when the counterfactual involves an unreachable set of values. Plausibility is one of the main focal points of this work.

3.3 Text Counterfactual Explanations

Counterfactual explanations for language models' classification of a sentence aims to present the user with a modified input sequence, which gives a different class upon subsequent classification by the model. Algorithmic recourse in the form of CEs for language model classification may not yet be a focus of research in the field; however, ensuring desiderata, like the plausibility of explanations, can benefit their quality. Apart from a generator's "label flip" objective, multiple other goals often exist.

Madaan, Saha, and Bedathur [42] treat text counterfactuals as adversarial examples or test samples that can break a model and subsequent fine-tuning with counterfactual samples as training against these types of failures. Wu et al. [71] argue that for some applications *label flipping* does not need to be the main objective. For example, same-label counterfactuals might serve as good training samples when generating counterfactuals for dataset enhancement. Furthermore, they suggest that a broad, more generic pool of counterfactuals can be preferable for an exploratory approach to model explainability. The authors of MiCE [56] stay true to the more original notion of CEs by constructing contrastive examples via minimal edits. They state that the perturbations should edit only a minimal amount of tokens, and the resulting counterfactuals should be fluent, "resulting in text natural to the domain".

There are various approaches to generating CE for language models, which we split into three main subclasses:

1. LLM-assisted generation

2. Latent perturbation and decoding
3. Sequential generation

Our categorization is motivated by the primary mechanism of text generation seen in text CE generators. We notice that most of the generators formulated in the literature adhere to at least one of those classes. We will use one representative model for each class as a baseline for the generator type.

3.3.1 LLM-Assisted Generation

LLM-assisted generation is the broadest category in our categorization. In this category, we capture all the methods that use an LM with text generation capabilities to produce parts of counterfactuals or whole counterfactual texts. Models that use different types of language models or different language model heads, such as masked language models, are excluded from this category.

The LLM-assisted generators exploit the large language model ability to model languages. The texts generated by those methods do not have to be constrained by the structure of the original sentences, thus enabling the generators to generate texts of high fluency.

The *Polyjuice* counterfactual generator [71] has been created to enable automatical counterfactual sentence generation, a previously very costly task due to mainly being done by human annotators. The authors aim to generate fluent and diverse counterfactuals by involving minimal necessary changes to the texts while giving control over the relationships between the factual and counterfactual sentences. The Polyjuice method was not initially created for label-flipping counterfactual text generation, with the authors focusing on general-purpose counterfactuals. However, the technique became one of the popular baselines for evaluating other text CE methods.

To train Polyjuice, the authors combine six sentence-pair datasets to form a train set containing close counterfactuals. In training, the sentence pairs are separated with a special token called control code to allow for the conditioning of the counterfactual in the original text. The authors list 8 control codes: `negation`, `quantifier`, `shuffle`, `lexical`, `resemantic`, `insert`, `delete`, `restructure`.

Because the Polyjuice method requires sentence-pair data with ground-truth perturbed sentences, in many cases, such as ours, it is impossible to fine-tune the model with task-specific control codes. In our evaluation, we use the eight control codes provided by the base version of the model.

The counterfactual generation in the Polyjuice method is performed by prompting the fine-tuned GPT-2 model consisting of three main parts: the factual sentence, the control code, and a masked factual sentence. Figure 3.1 illustrates an example prompt. The masked input can consist of multiple masks, each able to cover multiple tokens. The model outputs the tokens predicted for each provided mask, delimited with a special token `[ANSWER]`. The final counterfactual is recomposed by inputting the predictions into their respective mask locations.

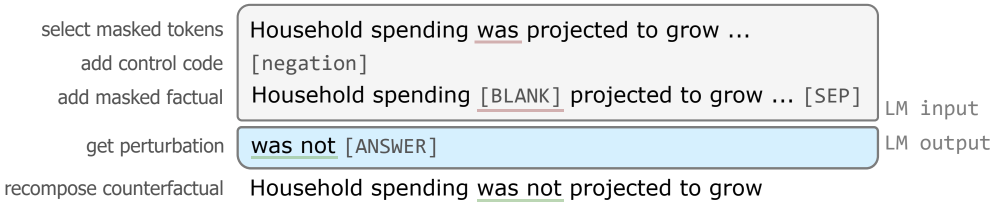

select masked tokens
Household spending was projected to grow ...
add control code
[negation]
add masked factual
Household spending [BLANK] projected to grow ... [SEP]
LM input
get perturbation
was not [ANSWER]
LM output
recompose counterfactual
Household spending was not projected to grow

Figure 3.1: An example counterfactual generation with the Polyjuice method. A prompt consists of a factual sentence, a control code, a masked factual, and model-specific special tokens. A single mask is used in this example; however, the model can handle an arbitrary number of masks.

3.3.2 Latent Perturbation and Decoding

Another type of text CE methods focuses on finding counterfactuals through latent perturbation. This concept is similar to the approaches used in counterfactual generators for tabular data, wherein the CE could be generated by finding the suitable perturbation in the data domain. Methods that we include in this category should perform some transformation of the text using a latent embedding, either through a simple search in the embedding space or a perturbation of the latent embedding using a surrogate model.

One of the most popular methods that implement latent perturbation is the *Plug-and-Play Language Models* (PPLM) method by Dathathri et al. [16]. PPLM enables text generation with parameter control by employing an auxiliary discriminator model. The gradients of the discriminator model are used to steer the text generation toward the target parameter or label while maintaining the fluency of the text using a main unconditional LM.

While PPLM is a method for controlled generation and was not made with counterfactual generation in mind, the central concept of the method allows for this type of usage. It has been adapted to be used in counterfactual generation in the past with methods like Generate Your Counterfactuals (GYC) [43] and Counterfactual Sentence Generation with Plug-and-Play Perturbation (CASPer) [42]. These methods extend PPLM from conditioning the text generation solely on an attribute, such as a class, to conditioning both on the attribute and a factual text.

PPLM formulation

PPLM models an input sequence embedding of a transformer LM as a history matrix $H_t = [(K_t^{(1)}, V_t^{(1)}), ..., K_t^{(l)}, V_t^{(l)})]$, where $(K_t^{(i)}, V_t^{(i)})$ is a key-value pair of the model's attention at the i-th layer, generated at time-steps 0 to t. Following this formulation, a transformer language model can be written as $a_{t+1}, H_{t+1} = LM(x_t, H_t)$, a model which, based on the input sequence and its embedding, generates the next token logit a_{t+1} and a history matrix H_{t+1} containing that token. To enable control of the text parameters, the generation step gets split into two parts: (1) Steering generation and (2) Ensuring fluency. In each part, the history matrix is shifted towards a gradient of a model in (1) an attribute model and (2) an unconditional language model. While in practice, the two optimizations are applied in a single step by combining the gradient information, they are illustrated in Figure 3.2 separately in red and blue, respectively.

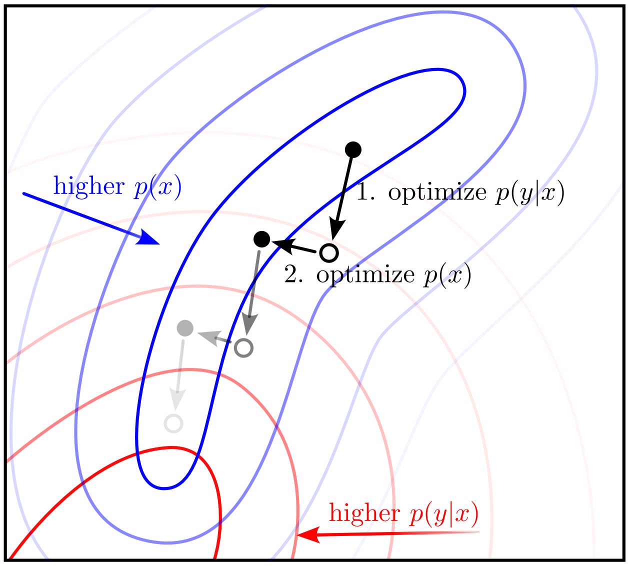


Figure 3.2: A simplified explanation of the optimization parts used in PPLM. The two regions represent the distributions of the attribute (red) and fluency (blue) models. The dots represent the embedding of the sequences at different steps of the generation process. After optimizing for the attribute model, the fluency of the text can be low, thus optimizing for fluency is necessary. In practice, the update is done in a single step by summing the two gradients. Adapted from Dathathri et al. [16].

Steering the text generation consists of using an attribute model $p(y|x)$ as a guide for the attribute adherence of the text. The authors explore two variants of an attribute model: a discriminator attribute model, for example, a fine-tuned LM with classification head, and a Bag of Words (BoW) model. The attribute model then gets formulated as $p(y|H_t)$, where y is an attribute (red in Figure 3.2). Since the goal is to perturb the history matrix of the model, we take a ΔH update matrix, initialized at zero, and updated with the gradients of the attribution model. Applying steps towards the positive gradient of the model for the given attribute updates ΔH_t:

$$\Delta H_t \leftarrow \Delta H_t + \alpha \frac{\nabla_{\Delta H_t} \log p(y|H_t + \Delta H_t)}{||\nabla_{\Delta H_t} \log p(y|H_t + \Delta H_t)||}$$

An updated history matrix $\tilde{H}_t = H_t + \Delta H_t$ can now be used in the language model to obtain the updated logits [16].

The second step of the method takes the updated history language model $LM(x_t, \tilde{H}_t)$ (blue in Figure 3.2) and transforms the perturbation to ensure a high level of text fluency. The motivation behind this step comes from previous work where the researchers noticed that tuning solely for a particular attribute model might quickly produce unrealistic and adversarial results. In this step, ΔH_t is tuned to minimize the KL-divergence between the updated $LM(x_t, \tilde{H}_t)$ and the base $LM(x_t, H_t)$ model.

Once the perturbation is completed, the resulting language model can perform inference on the modified $\tilde{H}_t$ to produce the next token a_{t+1}.

One of the main disadvantages of PPLM, addressed by Madaan, Saha, and Bedathur [42] and Madaan et al. [43] in GYC and CASPer, is the fact that the method is conditioned on an initial sequence of tokens rather than the full text. This means that to create a counterfactual, one needs to take a part of the factual sequence and input it into PPLM, which generates new text that adheres to a given (counterfactual) conditioning class without knowing the rest of the factual. This usually results in counterfactual sentences that stray from the initial text's topic or tone. We do not use GYC or CASPer as the authors do not open-source those models.

3.3.3 Sequential Generation

Counterfactual text generation does not always require significant changes in the input sequence. The sequential generation category uses this fact by reducing the generated words to only those chosen to be replaced. In this category, we include those methods that try to generate CEs by masking select parts of the input sentence and then infilling them with words that should change the predicted label of the resulting sentence. We do not limit the model types used in generating the new sentence parts, so overlap with other generator categories is possible, for example, in the case of Polyjuice [71].

An example method that we place in this subcategory is *Relevance-based Infilling for Natural Language Counterfactuals* (RELITC) [7] based on Minimal Counterfactual Editing (MiCE) [56]. RELITC's procedure of generating a counterfactual is illustrated in Figure 3.3. It uses an attribution method, such as Integrated Gradients, to generate token importances of an input for a fine-tuned discriminative LM. The token importances are calculated for either the target or the original label. RELITC masks $p\%$ of the tokens with the highest attribution scores, then infills those tokens sequentially using a Conditional Masked Language Model (CMLM), here a fine-tuned BERT model (introduced in Section 2.2.2). The optimal fraction of masked tokens $p\%$ is established through beam search. A candidate value $p \in (p_{min}, p_{max})$ is sampled, and several counterfactuals using this p value are generated. If the generation using this p value was successful and a counterfactual was classified as the target class, the value of p_{max} is set to the current p. If no successful counterfactual was found, the value of p_{min} is set to p. The procedure is then repeated, choosing new p, etc. until a depth limit is reached or the values of p_{min} and p_{max} are so close they do not affect the number of masked tokens in the factual.

The method's main contribution over MiCE is the inclusion of the infill order in counterfactual generation. The infilling order is established based on the uncertainty of the CMLM's predictions (steps 4 and 5 in Figure 3.3). For each of the masked token positions, the entropy of the output logits is calculated, and the position with the lowest entropy, thus the lowest uncertainty, is infilled.

The BERT model is fine-tuned on a training set using the masked LM task. As illustrated

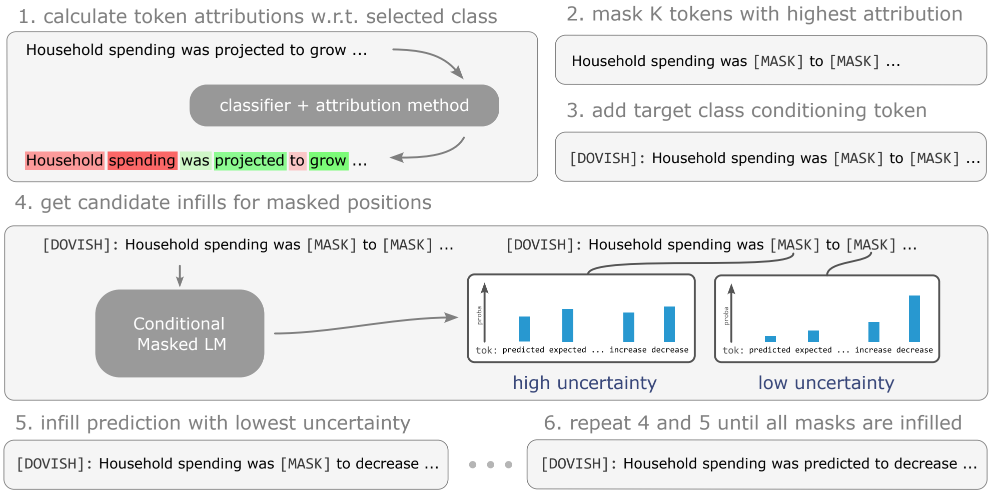

1. calculate token attributions w.r.t. selected class
Household spending was projected to grow ...
classifier + attribution method
Household spending was projected to grow ...
2. mask K tokens with highest attribution
Household spending was [MASK] to [MASK] ...
3. add target class conditioning token
[DOVISH]: Household spending was [MASK] to [MASK] ...
4. get candidate infills for masked positions
[DOVISH]: Household spending was [MASK] to [MASK] ...
Conditional Masked LM
[DOVISH]: Household spending was [MASK] to [MASK] ...
proba
tok: predicted expected ... increase decrease
proba
tok: predicted expected ... increase decrease
high uncertainty
low uncertainty
5. infill prediction with lowest uncertainty
[DOVISH]: Household spending was [MASK] to decrease ...
6. repeat 4 and 5 until all masks are infilled
[DOVISH]: Household spending was predicted to decrease ...

Figure 3.3: An example counterfactual text generation using the RELITC method.

in Figure 3.3, authors append the class label to the beginning of the training samples in the form of `LABEL : TEXT` to condition the model on the target label. The authors of BERT call this addition in the fine-tuning process a Conditional Masked LM.

3.4 Evaluating Text Counterfactuals

Evaluation of text counterfactual explanations generally employs both qualitative and quantitative metrics. In this section, we present how both kinds of metrics are used in the literature on text CEs: quantitative metrics in Subsection 3.4.1, and qualitative metrics in Subsection 3.4.3. We describe the quantitative metrics used in our work in greater detail, including their formulations and implementation details in Subsection 3.4.2. We use the metrics described in this section in the design of our experiments. The quantitative metrics are used in Experiment 1 (Section 4.2), while for Experiments 2 and 3 (Sections 4.3 and 4.4) we extend the definitions of two of the qualitative metrics.

3.4.1 Quantitative Metrics for Text CEs

Quantitative metrics used in CE evaluation are often used as a proxy measure of the quality of the generated texts. Since they can be computed relatively quickly and cheaply, researchers often use them to evaluate large amounts of text without needing expensive and time-intensive human annotation or evaluation. Quantitative metrics can reflect some of the desiderata set by the researchers in their research or method description.

One of the most frequent desiderata, *minimality* of the changes, is often measured using distance metrics, measuring the distance between the counterfactual and the factual texts. Some examples of these metrics and their usages are Levenshtein character edit distance [7, 20, 27, 56, 71], syntactic tree edit distance measures [27, 43, 71], or semantic measures of similarity, such as Universal Sentence Encoder [55] and embedding distance [7, 25, 42].

Another desideratum that researchers often opt to fulfill is the *validity* of the generated counterfactuals. This is frequently measured using the flip rate of the generated counterfactuals [7, 43, 55, 56, 71], or the average number of label changes towards the target class per counterfactual. Other approaches consider the classifier accuracy [64, 72, 74], or the label flip failure rate [22].

The main goal in text generation is ensuring the output sample is valid when interpreted by a human. Thus, another quality sought after in text counterfactuals is the *fluency* of the text. The metric most frequently used when quantifying the CE fluency is perplexity [16, 22, 42, 64] calculated as the average exponentiated negative log-likelihood of a sequence, most often computed using a GPT-2 model. The perplexity score has also been used as a measure of the *plausibility* of a text; Bhan et al. [8] compute the ratio between the perplexity of an initial text and its counterfactual examples as a means to compare the quality of the two texts.

Another metric widely used in evaluating text counterfactuals is Bilingual Evaluation Understudy (BLEU) [50], a metric used for assessing the similarity of two sets of texts, initially created for evaluating the task of machine translation. While in some works [7, 22, 72] the metric is used to measure the CE's similarity to factual sentences, a surprising amount of works uses it to quantify the *diversity* of the generations. Generating a set of diverse explanations for each factual sentence can be helpful when a counterfactual method aims to generate several explanations and then distill them using other metrics [7, 71], or when we seek to generate text samples for model fine-tuning [42, 71]. In the field of text counterfactual explanations, BLEU and its variations, like the self-BLEU [78], are used to measure the difference between the generated texts [20, 42, 43, 64, 71]. A lower value of self-BLEU is then sought after, as it signifies less similarity between the counterfactuals and, thus a higher diversity. Other approaches to diversity quantification include counting the distinct N-grams in the outputs [16].

3.4.2 Quantitative Metrics Implementation

Levenshtein distance [38], also known as edit distance, is a string similarity metric. For two strings, a starting string a and target string b, the Levenshtein distance consists of the sum of additions, deletions, and modifications needed to transform a to b. Initially introduced as a means of error correction in the field of coding theory, the metric has been adapted to many applications, such as string matching for optical character recognition systems [28].

The Levenshtein distance between two strings is given by:

$$Leven(a,b) = \begin{cases} min(|a|,|b|) & \text{if } |a| = 0 \text{ or } |b| = 0 \\ Leven(tail(a), tail(b)) & \text{if } head(a) = head(b) \\ 1 + min \begin{cases} Leven(tail(a), b) \\ Leven(a, tail(b)) \\ Leven(tail(a), tail(b)) \end{cases} & \text{otherwise} \end{cases} \tag{3.2}$$

Where the function $head(\cdot)$ acting on a sequence $x = \{x_1, x_2, ..., x_n\}$ returns the first element of the sequence, x_1, and $tail(x)$ returns all the elements except for the first one, $\{x_2, x_3, ..., x_n\}$.

This is a recursive definition of the metric. We use a more space-efficient implementation of the Levenshtein distance utilizing dynamic programming [28].

Syntactic tree distance is a metric for calculating the similarity between two trees representing sentences by counting the minimum number of node operations needed to transform a tree a to a tree b.

To calculate a distance between two trees, we use a tree distance algorithm called Zhang-Shasha algorithm [76], which, similarly to the Levenshtein distance, allows for node insertions, deletions, and modifications. In our evaluations, we use an implementation from the Python package `zss` [30].

To create the syntactic trees representing sentences, we use the functionalities of the Natural Language Toolkit (`NLTK`) [10] Python package. `NLTK` is a set of tools for handling natural language data, including parsers, tokenizers, and methods for part of speech tagging.

To compute the semantic tree edit distance from two sentences, we follow these steps:

1. Split sentences into tokens using `nltk.word_tokenize`.
2. Tag each token's part of speech using `nltk.pos_tag`.
3. Parse the tagged sequences into trees following a formal grammar using `nltk.RegexpParser`.
4. Transform trees into `zss` tree objects.
5. Get edit distance using `zss.simple_distance`.

While similar to the string edit distance, we expect tree edit distance to be more relevant to the task of counterfactual text generation. The string edit distance metric can be more sensitive to changes in individual words. However, in cases where the counterfactual generator masks and replaces whole words, the string edit distance can give different results depending on the length of the new token.

Embedding distance is the distance between two points in the high-dimensional representation space of a machine learning model. We choose the embeddings of the last layer of the `roberta-large` classifier as the representations of the evaluated sentences. For each counterfactual pair, we compute the Euclidean distance between the embeddings of the sentences.

Flip rate is simply the fraction of the counterfactuals classified to their target class by the classifier. For a model $f(\cdot)$ outputting a classification y_n for a sample x_n and a target class y'_n, the metric is calculated as follows:

$$\sum_{i}^{n} \frac{[f(x_i) = y'_i]}{n}$$

Where n is the total number of samples in x. The Iverson bracket, $[\cdot]$, returns 1 if the condition in the bracket is true and 0 otherwise.

Perplexity, the exponent of the entropy of a distribution is a measure of uncertainty. It was initially introduced to the field of language modeling by Jelinek et al. [33] as a general measure of the complexity of a language model. It has since been widely used as a main evaluation metric in comparing models' performance for the next token prediction task [39, 46].

For a language model f with a task of predicting the next token x_i for a sequence of tokens $X = x_1, ..., x_{i-1}$, the calculation of the perplexity metric assumes an approximation of the word error rate as the log-likelihood of the ith token conditioned on the previous tokens: $p_f(x_i \text{ is correct}) \approx \eta_1 \log p_f(x_i|x_{<i}) + \eta_2$ for some constants η_1 and η_2 [12].

We use the Hugging Face's `evaluate` [67] Python implementation of perplexity to evaluate the counterfactual sentences. The package uses the following definition of perplexity:

$$PPL(X) = \exp\{-\frac{1}{n}\sum_{i}^{n} \log p_f(x_i|x_{<i})\}$$

which for each token x_i in an input sequence of tokens $X = x_1, ..., x_n$ sums its negative log-likelihood conditioned on preceding tokens $x_{<i}$ before the exponentiation. The model used in the calculation of the log-likelihood is a GPT-2 large [54].

It is worth noting that perplexity is a metric for evaluating and comparing the fluency of language models. In text counterfactual generation, this metric is often used to represent the fluency of the counterfactual dataset itself, keeping model M the same while comparing different methods of generating counterfactuals. By doing so, the perplexity score obtained from this comparison relates to how likely it is for a model to have encountered a text like the one evaluated in its training.

Perplexity ratio is the ratio between the perplexity score of the factual and its counterfactual [8]. For each counterfactual method, we compute the mean of the perplexity ratios of its factual-counterfactual pairs. While the results of this metric might be closely dependent on the results of the perplexity metric, we expect that calculating the ratio for each factual-counterfactual pair can make the result less dependent on the absolute perplexity values.

3.4.3 Qualitative Metrics for Text CEs

Text explanations to natural language machine learning model classifications should exhibit qualities humans desire. Thus, human annotators are often asked to evaluate text counterfactuals. Many metrics used in text CE evaluation can be traced back to early works on machine translation evaluation [31, 70].

One of a counterfactual text's most frequently mentioned qualities is its *fluency*. This quality can be traced back to early works on machine translation that tried to unify what constitutes fluency in a machine-generated text. White, O'Connell, and O'Mara [70] describe fluency measurement as determining whether a piece of text "reads like good English", disregarding the semantic correctness of the sentence and giving it a rating on a n-point scale. At the same time, longer and more defined definitions exist, such as "A fluent segment is one that is grammatically well formed; contains correct spellings; adheres to the common use of terms, titles and names; is intuitively acceptable; and can be sensibly interpreted by a native speaker of English." by Ma and Cieri [41].

Many of the recent works on text CEs [7, 16, 43, 56, 71] evaluate their texts using a very similar notion of fluency as that defined by White, O'Connell, and O'Mara [70]. However, the notion of fluency has often been described vaguely or inconsistently. In several works, the annotators simply asked whether the Other works use different names like *naturalness* [55, 64] to measure essentially the same thing.

Another critical aspect of a text generated in this context is the semantic resemblance of the counterfactual to the original text. While we want our CE to be assigned a different label, we also should maintain the text's original meaning. This quality can be compared to the measure of *fidelity* in machine translation described by Hovy, King, and Popescu-Belis [31] as how well the "system's output text expresses the content of an equivalent portion of the source text". Similarly to the case of fluency, this early description matches multiple metrics used in the literature of text CE generators, such as *content preservation* [7, 43, 72] and *original content resemblance* [37].

Apart from the two main metrics, more specific ones are often used if the researchers want to test whether their method successfully employs certain desiderata or reaches particular goals. A notable example is the use of *plausibility* and *reasonability* by Yang et al. [75]. The method is evaluated on a rather difficult dataset of sentences in the financial domain of mergers and acquisitions. Here, the authors ask domain specialists to assess whether or not it is plausible to find sentences similar to the generated counterfactuals in their domain of expertise and if the changes made to the texts resulting in counterfactuals are reasonable.

The definition of plausibility outside of counterfactual explanations for language models often refers to the explanation's similarity or closeness to the original data distribution [35]. Indeed, many approaches to generating counterfactual explanations that emphasize the interpretability [65] or the robustness [5] of the explanations employ strategies that enhance the adherence of the counterfactual to a certain class.

Altmeyer et al. [3] define plausibility as:

> Let $\mathcal{X}|y^+ = p(x|y^+)$ denote the true conditional distribution of samples in the target class y^+. Then for x' to be considered a plausible counterfactual, we need: $x' \sim \mathbf{X}|y^+$.

We expand the definitions of fluency and plausibility for use in evaluations in Chapter 4.

Chapter 4

Experimental Setup and Evaluation Methodology

In this chapter, we describe the approach taken in our experimental setup and the evaluation methodology of the results gathered in our experiments. Section 4.1 describes the reasoning behind our choice of data and CE generators. Sections 4.2, 4.3, and 4.4 describe how we design and execute our experiments.

4.1 Task Description

The FOMC communications dataset [58] described in Section 2.3 gathers a wide array of communications released by FOMC labeled into three classes: dovish, hawkish, and neutral. The only preprocessing steps done by the authors of the dataset are splitting long sentences into segments using simple rules and labeling. Thus, the dataset offers a realistic representation of the data available to practitioners and researchers interested in using LM models for hawkish-dovish classification.

We assign target classes for the counterfactual generation to the original FOMC dataset (Section 2.3). We generate a random target class for each factual, keeping the original class priors. We share the generated dataset on the HuggingFace platform under `https://huggingface.co/datasets/karoldobiczek/fomc-communication-counterfactual`.

4.1.1 Model Selection

In Section 3.3 we introduce a categorization of text counterfactual generators into three categories: 1. LLM-assisted generation, 2. Latent perturbation and decoding, and 3. Sequential generation. To compare the different approaches to generating counterfactuals for LMs (RQ 2), we select a single model out of each category for our evaluations.

From the first category, we selected the Polyjuice method for our analysis because it has been widely used as a baseline for comparison with other generators. The ContRastive Edits with Sparse raTionalization (CREST) method [64] uses an approach similar to the one described in Section 3.3.3. Gilo and Markovitch [27] introduce a search-based approach for the counterfactual generation and justify the comparison with Polyjuice because both methods optimize for a similar goal – a small syntactic distance to the counterfactual. COunterfactual Generation via Retrieval and Editing (CORE) [20] trains a counterfactual retrieval process to provide factual samples to a GPT-3 model that generates the counterfactual. Madaan, Saha, and Bedathur [42] adapt Plug-and-Play Language Models (PPLM) (described in Section 3.3.2) using a GPT-2 model as a fluency guide. Both CORE and Madaan, Saha, and Bedathur [42] use a GPT model in generating the counterfactuals, making the comparison with Polyjuice natural, especially for the latter case.

Since the RELITC [7] (described in Section 3.3.3) model masks and infills tokens to generate counterfactuals (third category), its inner workings can be compared with Polyjuice, however, the way it performs those tasks is different. RELITC uses information from the task in two ways. First, it masks tokens using token-wise attributions from a trained classifier, and second, it uses a Conditional Masked Language Model to infill tokens based on the target class.

PPLM [16] uses the information about the classifier in a completely different manner. Conditioned on a starting sequence of words, it uses a language model to generate tokens that adhere to the target class by directly optimizing them with a trained classifier. We select PPLM as a representative model from the second category.

With this selection of counterfactual generators, we compare a method that generates perturbations with no data-specific goal, one that is informed by the data and the classifier but only perturbs single tokens, and one that generates text steered directly by the classifier.

4.1.2 Model Parameters

We use the `Polyjuice.perturb` method implemented in the Polyjuice package [1] to generate perturbations. The method is a wrapper that automatically generates and executes a perturbation prompt with the Polyjuice model. It accepts arguments like the number of perturbations, perturbation types, or custom masks. It also provides an option to generate random masks. For each factual, we generate 5 perturbations with random mask placement and random perturbations.

We use the implementation of PPLM provided by the authors [2]. We generate 5 counterfactuals per each factual sentence, limiting the generation to 25 tokens. As a fluency model we use the `gpt2-medium` and train a discriminator attribute model with the FOMC dataset (Section 2.3). As inputs to the method, we give a string of the first 20%, maximally ten words.

We use the authors' implementation of RELITC [3] and slightly modify it to use it with

[1] https://github.com/tongshuangwu/polyjuice

[2] https://github.com/nishthaa/PPLM

[3] https://github.com/Loreb92/relitc-counterfactuals

Input	Polyjuice	PPLM	RELITC
Accelerating productivity poses a significant complication for economic forecasting.	Accelerating productivity poses a significant complication for **the maintenance of financial stability**.	Accelerating productivity poses a significant complication for economic forecasting. **the key question for policymakers today is how much to invest in new and improved infrastructure to improve the efficiency and productivity of the economy**	Accelerating productivity poses a **further** complication for economic forecasting.
Market-based measures of inflation compensation remained low	Market-based measures of inflation compensation remained **high**	Market-based measures of inflation compensation remained low **in august, while the consumer price index (cpi), a measure of overall consumer prices, was up 0.1 percent**	Market - based measures of inflation **unemployment** remained low
These participants cited, for example, the still-elevated levels of long-term unemployment and workers employed part time for economic reasons as well as low labor force participation.	For example, **there were 65 members in the first year, 91 members at the end of the second year and 106 members in the third year**.	These participants cited, for example, the still-elevated levels of long-term unemployment and workers**’ comp that have led to a significant number of workers not being paid for time that they actually spent on job training or continuing**	These participants cited, for example, the still - elevated rates of long - term unemployment and **being** employed part time **slowing** economic **recovery** as well as **continuing** labor **market contraction**.

Table 4.1: Sample counterfactuals. The task of the first two rows of counterfactuals was changing sentiment from *dovish* to *hawkish*, while the third row is *dovish* to *neutral*.

multiclass classification problems. We calculate token-wise feature importance w.r.t the original label using Integrated Gradients. In generating counterfactuals, we use the default settings, infilling based on the highest model confidence.

In our experiments, for each method we use a single counterfactual explanation. Since for each factual, the generators generate a number of counterfactuals, we select the one with the highest probability when classified by the classifier to the target class. If none of the counterfactual variants are classified as the target class, we select one at random. Table 4.1 presents a sample of counterfactual explanations generated by all three methods using those settings. We share the full test set and the counterfactuals selected for the survey under `https://huggingface.co/datasets/karoldobiczek/fomc-communication-counterfactual` as well as the repository containing the code used to generate the data under `https://github.com/drobiu/LMCounterfactualExplanations`. Due to the long inference time, PPLM counterfactuals were generated on the DelftBlue supercomputer [17] using an A100 GPU.

4.2 Experiment 1: Quantitative Metrics

As discussed in Subsection 3.4, many methods for generating text counterfactual explanations are designed in a way that allows them to employ certain desiderata. Methods are designed to optimize some objectives, either intentionally or implicitly. Then, metrics measuring the desired objective are used when evaluating their method. Due to that, comparing with baselines that do not employ those particular objectives is often complex or unrealistic.

Our first experiment evaluates the text counterfactual generators using quantitative metrics. We select a variety of commonly used metrics in text CE evaluation, intending to create a comparison that is not biased by using a single type of metric. Using the results of this experiment we want to contribute to answering RQ 1 and RQ 3.

For this experiment, we select both minimality and fluency metrics. For the metrics measuring the minimality of the changes, we choose the Levenshtein edit distance, the semantic tree distance, and the embedding distance. To measure the fluency of the counterfactuals, we measure the perplexity of the explanations and the perplexity ratios between the explanation and the factual sentence. We report an average computed over all samples in the test set for each metric.

The metrics we selected are commonly used to evaluate text counterfactuals in the literature. However, due to our task formulation, we rejected some of the other popular metrics. Since our dataset does not contain factual-counterfactual pairs, we do not evaluate the generators using the BLEU metric, as it requires at least one ground truth (in this case, a counterfactual) sentence. We do not measure the self-BLEU metric, commonly used in quantifying the diversity of the counterfactuals, since we only present the annotators with a single counterfactual per factual.

4.3 Experiment 2: Fluency of the Explanations

An essential quality of a counterfactual explanation of a text is the quality and soundness of the explanation in terms of its fluency. Similarly to counterfactual explanations in other data types, it has to match the factual domain for an explanation to be valuable and give insight into the model and its decision-making. An explanation that solely optimizes the label-flip objective can often look like an adversary example, fooling the classifier model and bringing little to no information about the model itself.

In the second experiment, we want to verify which of the generators generates texts of high fluency, gathering data that will contribute to answering all three research questions (RQ 1, RQ 2, and RQ 3). Thus, we evaluated the explanations by letting human annotators judge the counterfactuals' fluency. We extend the fluency metric defined in Section 3.4.3 to establish a concrete definition to be used by the annotators.

4.3.1 Defining Fluency

In designing a task for human evaluators, it is necessary to consider how they interpret the task's prompts. Especially in a field like text interpretation, non-experts can understand a value like fluency in many different ways. Not providing a definition or using a very broad one may lead to annotators essentially evaluating different qualities. It is thus crucial to establish a robust and detailed definition upfront.

In Section 3.4.3, we describe how fluency has been defined both in the scope of text counterfactual explanations and in fields like machine translation. Here, we derive a fluency definition by modifying one by Ma and Cieri [41].

The generators we use can produce texts where word capitalization is omitted or where the text changes abruptly. This impacts the quality of the generated text. To omit ambiguity in case a counterfactual contains these errors, we specify that they will also impact fluency. Our final definition is as follows:

> A fluent segment is one that is grammatically well-formed; contains correct spellings; adheres to the common use of terms, titles and names; contains properly capitalized letters; and is intuitively acceptable. Unfinished sentences also impact the fluency of a segment.

We ask the human annotators to judge the fluency of each sentence on a scale from 1 to 5, with 1/5 being *very bad*, 2/5 being *bad*, 3/5 being *sufficient*, 4/5 being *good*, and 5/5 being *very good*. With this range of grades, we aim to allow the annotators to mark a sentence as average but also allow them to grade exceptional sentences high.

In addition to the grade labels, we give the evaluators guidelines for the *very bad*, *sufficient*, and *very good* fluency grades:

> A text should receive a score of:
>
> - 5/5 if it follows this definition completely.
> - 3/5 if there are several mistakes but the text still is interpretable.
> - 1/5 if it is not fluent or grammatically correct English.

4.3.2 Survey Design

We construct our study as a survey using the Qualtrics[4] platform. We obtain a TU Delft Human Research Ethics (HREC) approval to conduct our study. To proceed to our survey, each participant has to agree to an informed consent form specified in Appendix A as a part of the HREC procedure. The concept of counterfactual explanations can be novel and

[4] `https://www.qualtrics.com/`

complicated to some people, especially for those unfamiliar with machine learning. We provide a short description of counterfactual explanations to introduce the participants to the study's purpose and domain. We also introduce the task which the annotators will perform in the survey. Finally, we describe the fluency metric and grading criteria, as introduced in Subsection 4.3.1. We include the complete introduction in Appendix B.

In the survey's main part, we ask the participants to rate the counterfactual explanations using our fluency definition and scale. We create a separate survey page for each factual sentence containing all three of its counterfactual counterparts and fields for inputting their fluency scores. Since, in this experiment, we do not recruit expert annotators, we do not disclose the classes of the factual and counterfactual sentences. By doing so, we aim not to distract the annotators with information that is not relevant to their task, especially when the classes themselves would require additional explanations.

Finally, each participant randomly receives a fixed number of factual-counterfactual pages. To minimize the likelihood that the order in which the counterfactual sentences are displayed influences the rating given to the sentence by the participant, we also decided to generate all possible permutations of the counterfactual sentences and display them randomly.

4.3.3 Participant Recruitment and Sample Sizes

We use the Prolific[5] platform to recruit human annotators for our evaluation task. We recruit native English speakers from the United Kingdom and the United States of America. The participants are prescreened so that only those with at least high school-level education can participate in the study. The annotators are compensated with the standard for Prolific rate of 9 GBP per hour.

With our fluency evaluation instructions, we try to make the grading process nonsubjective, yet some variance in the gradings is expected between annotators. Thus, we chose a sample size of 5 gradings for each counterfactual sentence. We decided to grade the counterfactual sentences of 100 samples from the dataset, which, with three counterfactual generators, gives a total of 300 graded sentences.

Apart from the amount of grade samples needed per counterfactual sentence, we also consider the time necessary to complete a full survey. The longer the survey duration, the more likely the respondent will lose interest in it and start to give intentionally low-effort responses. We thus run pilot studies with small groups of respondents, varying the number of counterfactuals they grade to determine the time it takes on average to complete our surveys. Apart from the survey completion times, we analyze individual respondents' average grades per pilot study batch. This determines whether the amount of factual-counterfactual pages in a survey impacts how the respondents grade the sentences. We discuss this analysis and the full results of our study, which consists of the combined results of the five pilot studies and the full-scale study in Chapter 5.

[5] `https://www.prolific.com/`

4.4 Experiment 3: Expert Evaluations

Our primary goal in this work is to determine the usability of counterfactual explanations in financial text classification (RQ 2). Thus, we decided to perform another series of evaluations with domain experts. Including experts who deal with texts in the economic domain daily allows us to judge the texts' fluency closely and determine whether the changes introduced to produce the counterfactuals make sense given the original and target classes.

Any fluent speaker of a language can assess the fluency of a text with some level of certainty. Still, when evaluating specific domains with uncommon terms and jargon, regular language users can lack the knowledge necessary to identify word usage, which would decrease a text's fluency level. In the classification task of the dataset we analyzed, changing a single word can make the difference between a dovish and a hawkish sentence and thus cause the rest of the text to be incorrect or seem out of place. These changes might not always be caught by a typical language user and require additional expert analysis.

In a specialist domain like financial text classification, generating text sound solely in fluency is insufficient. In Section 3.4.3, we describe the different desiderata present in previous work on text counterfactual generation. Many approaches for evaluating counterfactual text generation techniques consider how well the approach preserves the original topic of the factual text and how realistic the changes or the resulting sentences are.

4.4.1 Defining Plausibility

In our expert evaluation, we, too, measure the quality of the counterfactual texts with regard to the domain we analyze. Similarly to Gilo and Markovitch [27], Madaan et al. [43], and Yang et al. [75], we ask the experts to judge the counterfactuals' plausibility. However, we decided to model our notion of plausibility to closer reflect the definition used in the literature on counterfactual explanations. We discuss the definition of plausibility by Altmeyer et al. [3] in Subsection 3.4.3.

We adapt this definition to the domain of explanations for text inputs and arrive at this definition:

> A plausible counterfactual segment adheres well to samples seen in the real data distribution, and the target sentiment of the target class. The changes made to the factual, considering the meaning and context of the edited words, should also fit the target domain.

The main goal of the definitions of our metrics is to guide the human evaluators to grade the characteristics of the counterfactuals we want to analyze. So, in the second part of our definition, we additionally ask the evaluators to focus on the change that the counterfactual introduces compared to the original sentence. This is inspired by the *reasonability* metric used by Yang et al. [75]. With this condition, we aim to allow the evaluator to consider the finished explanation but also entice them to judge the process or direction taken in generating it.

Similarly to the fluency metric, the evaluators receive additional guidelines for grading sentences using the plausibility metric:

> A text should receive a score of:
>
> - 5/5 if it follows this definition completely.
> - 3/5 if there are several mistakes but the text reflects the right sentiment.
> - 1/5 if the changes are nonsensical.

4.4.2 Survey Design

For each counterfactual, the survey page consists of three elements:

1. 1 to 5 scale: Fluency grading
2. 1 to 5 scale: Plausibility grading
3. Open question: Remarks about the counterfactual

Similarly to the design of the non-expert survey described in Subsection 4.3.2, we use the Qualtrics platform to build and host our surveys. We ask the annotators to grade the counterfactuals' fluency and plausibility. We do not modify the definition of fluency for this round of evaluations. We again use a 1 to 5 scale with labels for the grading.

Contrarily to the non-expert evaluation, we inform the experts about the texts' original and target classes. This information is crucial for grading the plausibility of the explanations. Since the experts are familiar with the texts and classifications in this domain, this addition should not cause unnecessary strain or dissociation for the participants.

To engage our evaluators even further in grading the counterfactual explanations and elicit more insights, in the final step of grading a counterfactual, we ask them to describe any remarks they might have about the text in an open question. The answer to this question is optional and is intended to be used when the evaluator sees shortcomings in the counterfactual they analyze. The following is the full text we present in the question:

> Considering the counterfactual from the previous question, describe what qualities that you might look for in a text like this are missing. Your comment can refer to the semantics of the sentence, its structure, or contents. If you do not have any comments you can also leave the answer empty.

Each page of the expert survey contains nine questions: six closed grading questions and three open ones. This format allows us to gather more information but makes the survey more time-consuming. We take this into consideration when adjusting the number of pages displayed to the evaluators.

4.4.3 Participant Recruitment and Sample Sizes

To maintain high quality and confidence in the ratings, we gathered ratings from specialists experienced in assessing and analyzing texts in the monetary policy domain. We contacted employees of three central banks and recruited 9 participants for the survey. The identities of the participants were pseudonymized and will not be disclosed.

The tasks we present to the evaluators in this survey require more time for consideration and are more complex than those in the non-expert survey. Thus, we decrease the number of pages in the survey to three. This allows us to receive 27 separate factual-counterfactual ratings. Since we maintain the number of separate ratings per factual-counterfactual instance at 5, we can grade 5 in full in this experiment.

While getting a single grade sample for each factual-counterfactual instance could allow us to receive gradings for a considerable part of our dataset, we instead chose to average ratings and remarks collected from multiple experts.

The median time to complete the survey was 17 minutes 36 seconds, while the average after removing extreme outliers was just over 20 minutes.

Chapter 5

Experimental Results and Discussion

In this chapter, we discuss the results of our experiments and their implications in the broader field of explainability for LLMs. We first analyze whether the survey duration affects the quality of the results in Section 5.1. Later, in Section 5.2, we present the results of our experiment. We analyze them deeper by testing the correlation between qualitative and quantitative metrics in Section 5.2.1. In Section 5.3, we analyze the expert comments on the counterfactuals.

5.1 Survey Duration

Section 4.3 describes our approach to building the survey for our second experiment. The time a person spends in one survey could negatively impact the outcomes. To establish an optimal amount of pages presented to the respondents, we try five different batches of surveys with differing amounts of pages.

We prepare surveys with 7, 10, and 15 pages and send them to 5 respondents. The average time to complete the surveys is 10 minutes 36 seconds for 7-page surveys, 10 minutes 12 seconds for 10-page surveys, and 12 minutes 32 seconds for 15-page surveys. The resulting timings of our pilot surveys show that running a full-scale survey with 15 factual-counterfactual pages could be feasible. However, due to inconsistent timing results in the latter two pilot studies and the preference for high-effort responses, we settled on a conservative 10-page survey. The full-scale survey of 25 respondents gave a median completion time of 10 minutes and 16 seconds.

With these small sample sizes of 5 respondents in each pilot study, we cannot draw firm conclusions, although we can determine whether any individual respondents give very similar responses or whose responses are very different from the main body of respondents. We calculate the average grades each respondent gave during the study. We then calculate

Batch (no. pages)		**1** (7)	**2** (10)	**3** (15)	**4** (15)	**5** (15)	**6** (10)
Intra-rater	Mean	3.10	2.92	3.31	3.55	3.24	3.15
	Std.	0.21	0.23	0.24	0.38	0.35	0.51
Inter-rater	Mean	3.10	2.91	3.31	3.74	3.41	3.14
	Std.	0.96	0.85	0.68	0.51	0.98	0.85

Table 5.1: Means and standard deviations of the intra-rater and inter-rater scores in each batch of surveys.

the mean and standard deviation of these ratings to obtain the mean *intra-rater* ratings. We present those in rows 2 and 3 of Table 5.1. We also report the *inter-rater* scores.

The standard deviations of the respondents' intra-rater for the first three pilot studies start out at around 0.2 and increase for the later batches due to the larger group of annotators. For the inter-rater means, we see a slight decrease in the average grade standard deviation in the first three batches, corresponding to the pilot studies with 7, 10, and 15 factual-counterfactual pages. This could signify a reduction in the quality of the responses caused by the decreasing patience of the respondents over time; however, in each pilot study, we give the respondents different factual-counterfactual samples. Samples for which the respondents are more certain could explain the lower standard deviation. Generally, we do not see a cause to reject the results of our surveys.

5.2 Experiment Results

The results of the quantitative metrics are presented in Table 5.2. The results do not seem to point toward a method that performs the best out of the three, although specific patterns emerge. PPLM, which uses a GPT-2 model in its generation phase and optimizes for its fluency, performs best for perplexity-based metrics. Similarly, RELITC, which tries to minimize the fraction of perturbed tokens in creating a counterfactual, has the best results for the Levenshtein distance and the tree edit distance. It also achieves the highest flip rate out of the three methods. Interestingly, Polyjuice achieves the best results solely for the embedding distance metric. These results are computed for all counterfactuals, including ones that do not succeed at flipping the label; however, comparing those to ones computed using successful CEs only in Table C.2, we see no major differences.

Generator	Perpl. ↓	Perplexity ratio	Edit dist. ↓	Tree dist. ↓	Emb. dist. ↓	Impl. ↓	Succ. rate ↑
Polyjuice	90.98 (172.1)	1.80 (4.6)	0.31 (0.3)	19.67 (24.0)	**20.32** (3.7)	33.64 (4.6)	0.34 (0.5)
PPLM	**36.97** (16.9)	**0.78** (0.5)	0.69 (0.5)	36.94 (10.3)	20.88 (3.7)	**32.18** (4.0)	0.51 (0.5)
RELITC	100.94 (125.2)	1.67 (1.2)	**0.14** (0.1)	**10.72** (12.2)	21.96 (3.9)	33.30 (3.9)	**0.74** (0.4)

Table 5.2: Averages and standard deviations of the quantitative metrics calculated for counterfactual explanations of texts in the test set. A perfect result for the perplexity ratio metric is thought to be 1 [8].

The results of the human-annotated qualitative metrics are displayed in Table 5.3. Even though the RELITC generator receives some of the worst results for the perplexity metrics,

	Annotators			
	Non-expert	Non-expert (5 CEs)	Expert	
Generator	Fluency	Fluency	Fluency	Plausibility
PPLM	2.86 (0.7)	2.48 (0.5)	2.26 (0.5)	1.83 (0.3)
Polyjuice	3.40 (0.9)	3.44 (0.7)	3.45 (0.9)	**2.45** (0.7)
RELITC	**3.43** (0.8)	**3.96** (0.5)	**3.90** (0.6)	2.12 (0.3)

Table 5.3: Results of the human annotation of the counterfactuals using the qualitative metrics. Each counterfactual receives five ratings. Those ratings are averaged. We display the averages of those averages and their standard deviations. Since the expert evaluations are performed on a subset of five samples, we calculate the fluency scores the non-expert annotators give on the same set of samples.

it produces the most fluent texts according to both the sampled groups. At the same time, the opposite applies to the PPLM method. Surprisingly, the only metrics that reflect the same ranking of the methods as the fluency are the Levenshtein and tree edit distances. These findings suggest that perplexity might not be a reliable metric for evaluating CEs (RQ 3).

All the methods receive rather low scores of the expert-judged plausibility metric, all achieving less than sufficient (score of 3, see Section 4.3.1) scores. Even though the RELITC method produces the most fluent counterfactuals, the experts assign the highest plausibility scores to Polyjuice. This stems from the RELITC's misuse of domain-specific words, as reported in the experts' comments analyzed in Section 5.3.

A surprising result is the closeness between the experts' and non-experts' grading of fluency. The highest difference between the average grades in Table 5.3 is 0.22 for PPLM, while Polyjuice's fluency scores differ by only 0.01. This indicates that the fluency metric might not be dependent on the annotator's background and that non-experts' ratings can give reliable results even in specialist domains.

Since the perplexity metric is highly dependent on the training data of the model used to compute it [46], we investigate whether the choice of models affects our results. We choose two different models apart from GPT-2, Open Pretrained Transformer (OPT) [77] opt-125m, and a GPT-2 model fine-tuned on four financial datasets. We present the resulting perplexity and perplexity ratio results in Table C.1; however, apart from different scales of perplexities, we notice no major differences between the results.

5.2.1 Agreeance Between the Results

Even though the expert and non-expert fluency scores are nearly the same and they dictate the same hierarchy as the distance metrics, we find that there is little apparent correlation between the qualitative and quantitative results.

We decide to analyze the agreeance between our results further. We perform statistical tests comparing the quantitative and qualitative metrics. We use the Pearson correlation coefficient to measure the dependence between metrics.

Agreeance with Non-Expert Ratings

	Perplexity	Perplexity ratio	Edit Dist.	Tree edit dist.	Emb. dist.	Implausibility
Correlation	-0.06 (0.2)	-0.03 (0.5)	**-0.21 (0.0002)**	**-0.21 (0.0003)**	0.03 (0.7)	0.06 (0.3)

Table 5.4: Pearson correlation coefficients between the quantitative metrics and the non-expert fluency results. Numbers in brackets represent the p-values.

Table 5.4 shows that there is no strong correlation between the perplexity metric except for the edit distance metrics. The correlation of fluency scores with both the Levenshtein edit distance and tree edit distance metrics show low p-value, below 0.005, suggesting a significant (negative) correlation with a coefficient of -0.21. This result is not surprising considering the earlier findings, which suggest that the methods that introduce a lower amount of edits tend to be rated higher.

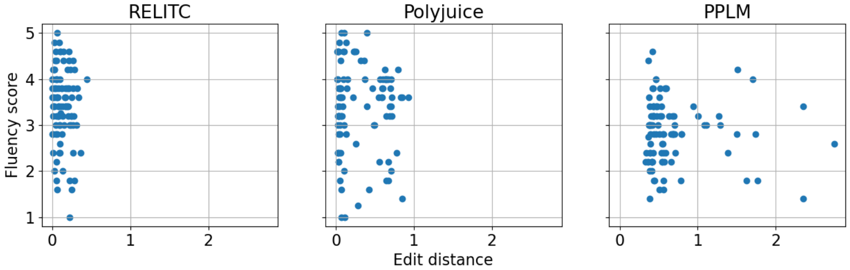


Figure 5.1: Scatterplot of the fluency score of a counterfactual by its edit distance.

To investigate this correlation further, we calculate the correlation coefficient between fluency and perplexity for each generator separately. We obtain the following results: Polyjuice: $r = -0.03$, p-value $= 0.73$, PPLM: $r = -0.06$, p-value $= 0.54$, and RELITC: $r = -0.11$, p-value $= 0.28$. These correlation coefficients suggest no statistically significant correlation between the edit distance metric result and the fluency scores. In Figure 5.1, we plot the distributions of the scores in relation to the edit distances. The plot shows that for each method, there is an initial cluster of low edit distances and a wide distribution of fluency scores (close to 0 edit distance for RELITC and Polyjuice and close to 0.5 for PPLM). There also exists a trend where the fluency scores decrease for the higher values of distances. This means that edit and tree edit distances can be good measures for assessing the fluency of text counterfactuals (RQ 3).

The only metrics that show a strong correlation with the non-expert fluency scores are the edit distance metrics. Our results show that for high amounts of changes made to the factuals, the fluency of the counterfactuals decreases. This holds for comparisons between counterfactual generators. However, there is no strong correlation between the two variables within the classes. Those two results suggest that the distance metrics are well-suited for comparing different counterfactual generators. One can also argue that those metrics can also be used within classes as a means of detecting outliers, as it is likely that counterfactuals with very high edit distances can be of poor quality.

Agreeance with Expert Ratings

	Perp.	Perp. ratio	Edit Dist.	Tree edit dist.	Emb. dist.	Impl.
Fluency	0.12 (0.6)	0.14 (0.6)	**-0.56 (0.016)**	**-0.56 (0.015)**	-0.25 (0.3)	0.13 (0.3)
Plausibility	0.32 (0.2)	0.02 (0.9)	-0.12 (0.6)	-0.28 (0.3)	-0.12 (0.6)	0.28 (0.3)

Table 5.5: Pearson correlation coefficients between the quantitative metrics and the fluency and plausibility scores given by the expert evaluators. Numbers in brackets represent the p-values.

Similarly to the analysis performed in Subsection 5.2.1, we calculate the correlation coefficient for the fluency and plausibility scores given to the sentences by the expert evaluators. Table 5.5 shows the resulting correlation values. Similarly to the non-expert correlation scores, we see a negative correlation between the edit distances and the fluency of the counterfactuals. The weak correlations are likely to stem from the low number of responses we acquired.

5.3 Expert Insights on Counterfactuals

As a part of our expert evaluation questionnaire described in Section 4.4, we ask our respondents to elaborate on the shortcomings they identify in the counterfactuals. We perform further analysis of the counterfactual generators using those comments. In the question we pose to the evaluators, we ask them to comment on shortcomings in "*the semantics of the [counterfactual] sentence, its structure, or content*". We gathered 50 comments, 15 on Polyjuice, 18 on PPLM, and 17 on RELITC counterfactuals. We present a factual-counterfactual sample with corresponding comments in Table 5.6. We observe distinct patterns for each of the generators.

Over half of the comments regarding Polyjuice counterfactuals relate to the lack of relevance of the introduced changes. The experts say that "*the content and hence tone has not changed at all*" or that with the changes that Polyjuice introduces, "*the sentiment has not been changed at all*". A large part of the comments address the grammatical errors in the Polyjuice counterfactuals: "*The only thing that has changed is the introduction of grammatical errors ...*". For one of the counterfactuals, the commenters notice that there is an "*... entirely different subject*" that replaces the original one.

As discussed in Section 4.2, PPLM generates tokens until reaching a fixed limit. Therefore, it is possible that the generator does not finish a sentence within the given limit. Thus, it is natural that the main point of critique offered by the experts is that the counterfactuals are unfinished. Similarly to Polyjuice, PPLM introduced errors in the sentences: "*The counterfactual didn't change the [tone], but included orthographic errors.*" However, unlike Polyjuice, PPLM's propensity to use domain-specific words introduces more room for errors in the usage thereof. This is reflected by the comments: "*It is odd to call something subdued when it is at a three-year high unless some caveats are given about the longer-term trend.*" PPLM was also criticized for making the counterfactuals too conversational.

RELITC is similar to PPLM in that it learns the domain-specific terms through its CMLM

	Text	Expert comments
Factual	At the conclusion of this discussion, the Committee voted to authorize and direct the Federal Reserve Bank of New York, until it was instructed otherwise, to execute transactions in the System Account in accordance with the following domestic policy directive: The information reviewed at this meeting suggests that the expansion in economic activity is still robust.	
Polyjuice	At the conclusion of this discussion, the committee voted to authorize and direct the federal reserve bank of new york, until it was instructed otherwise, to execute transactions in the system account in accordance with the following domestic policy directive: the information **was not** suggests that the expansion in economic activity is still robust.	**1:** "Language is off. The negation at the end makes the statement unclear.", **2:** "Again all capital letters are missing. This time, the last sentence is also incorrect"was not suggests" is clearly a mistake". This mistake makes the whole message impossible to understand.", **3:** "The last clause is not grammatically correct. Otherwise it does come across a bit more dovish."
PPLM	At the conclusion of this discussion, the committee voted to authorize and direct the federal reserve bank of new york, until it was instructed otherwise, to execute transactions in **securities that are not covered by the exchange act**.	**1:** "There now is a completely different meaning at the end of the statement.", **2:** "Again capital letters are missing, and the second sentence is incomplete. But at least the first sentence can be understood and sounds dovish (execute transactions in additional securities)", **3:** "There is an incomplete sentence at the end of the excerpt. It also loses the link to the current state of the economy and so isn't more dovish"
RELITC	At the conclusion of this discussion, the committee voted to authorize and direct the federal reserve bank of new york, until it was instructed otherwise, to execute transactions in the system account in accordance with the following domestic policy directive : the information reviewed at this meeting suggests that the **impact of the response** is still robust.	**1:** "There is a change of meaning in the last sentence which makes it less clear.", **2:** "All capital letter are missing, but the rest of the text seems to be correct. In terms of content, it is not clear at all, in particular the sentence "the impact of the response is still robust".", **3:** "The vagueness of 'impact of the response' makes it difficult to extract the message or signal this would try to send."

Table 5.6: Sample counterfactuals and the expert comments regarding them. Factual label: *neutral*, target label: *dovish.* Changes introduced in the counterfactuals, except for word capitalization, are **highlighted**.

and then uses them to generate a counterfactual. This, also similar to PPLM, introduces room for error. The experts comment on sentences where RELITC introduces domain-specific terms that are factually incorrect, contradict the contents of the sentence, or make the tone of the counterfactual unclear or conversational. Three of the comments mention that the counterfactuals might be pointing towards incorrect class, e.g., making the sentence more hawkish instead of dovish. Similarly to Polyjuice, RELITC receives many comments stating that the counterfactuals "*not seem different in tone.*"

The generators received minor comments about topic changes and incorrect or lacking capitalization. For two of the generators, Polyjuice and RELITC, some experts expressed that the counterfactuals might reflect the sentiment of the target class. In conjunction with the fact that Polyjuice receives considerably fewer negative comments about the incorrect usage of terms than the other two generators, this might explain the plausibility score results in Table 5.3.

5.4 Data Complexity

The complexity of the dataset a model is trained on can be a limiting factor in generating plausible explanations. Approaches such as CLUE [4] and REVISE [34] discussed in Section 3.2 employ surrogate models that can be used to estimate the data distribution closely to generate plausible counterfactuals. The same might apply to counterfactual generation for language models.

The relatively noisy results of the embedding-based metrics can suggest that the data is too complex for the model we are trying to explain. The embedding distances are calculated by comparing the distances between the last-layer embeddings of the classifier. If the classifier is not fitted well enough to the data, its embedding space might be too noisy with clusters of different classes overlapping. The LM counterfactual generators that we use might not be complex enough to generalize to the task and thus cannot generate plausible counterfactuals. However, another possibility is that the classifier is so noisy that even an adversarial-looking counterfactual can cause a label flip.

In our case, the classifier might indeed be underperforming. The authors of the model report a test set accuracy of 0.71 [58], a result that is rather low even considering some of the results reported in the original RoBERTa paper [39]. This result gives another use case for text CEs, raising concerns about the model's accuracy and possibly leading to questions about its usability.

One could argue that to estimate a more complex domain, we need a larger language model than those used in the generators we analyze. Approaches that use LLMs such as GPT-3 exist, [20] use a RAG approach to retrieving texts and editing them to generate counterfactuals, and Chen et al. [13] use GPT-4 to investigate *counterfactual simulatability*, whether an input's explanations allow inferring outputs for other inputs. Even in the domain of hawkish-dovish classification, there exist works that prompt an LLM to explain its classification [29].

We investigate this argument and formulate a pseudo-RAG counterfactual method (Appendix D) by prompting a GPT-4o model to generate a counterfactual with minimal changes to the input using supplied sample texts. Since the FOMC dataset has no factual-counterfactual pairs, we present random text samples from both the factual and counterfactual classes. We run the method on a subset of 15 factuals in our task dataset and present the quantitative results in Table D.1. The method achieves a very high success rate of 0.88, but it receives poor results for the edit distances and the implausibility metric, its counterfactuals staying far from the factual embeddings. A method like this does not use the classifier in the generation, and while methods like these can achieve good plausibility (ex. Polyjuice), this can inhibit its performance.

Chapter 6

Conclusions and Future Work

In this chapter, we summarise our work and contributions (Section 6.1) and form suggestions and recommendations for practitioners aiming to use LM counterfactual explanations in specialist fields (Section 6.2) and for future research (Section 6.3). We also state the limitations of the current work (Section 6.4).

6.1 Conclusions

Language model counterfactual explanations help analyze and explain natural language processing models. These explanations can ensure trust in applications of those models in specialist domains, such as finance, where the models can have high stakes in correctly understanding sentiments. However, unlike the usual tasks a LM counterfactual generator is tested on, a specialist dataset, such as one with a bank communications classification task, can prove challenging for both a LM model and a counterfactual generator. In this thesis, we set out to explore how counterfactual explanation generators for language models fare on a dataset from a financial domain and assess their usefulness for practitioners.

To aid in this assessment, in Section 1.2, we define the following research questions:

- *RQ1: What are the essential desiderata of text counterfactual explanations for language models?*
- *RQ2: What type of explanations is most useful for the practitioners?*
- *RQ3: Are the metrics commonly used in Natural Language Processing right for evaluating and comparing text counterfactual generation methods?*

To analyze the CE generators, we set out to explain a model trained on a novel dataset for sentiment analysis of central bank communications. We review various LM counterfactual generators, and based on the way they generate text, we derive three distinct categories described in Section 3.3 (later used in RQ 2).

In reviewing the literature on LM counterfactual explanations, we find various desiderata taken in their design (RQ 1) and ways the researchers evaluate their methods (RQ 3). We find that in previous work, researchers often focus on evaluation through quantitative metrics and neglect explicitly defining qualitative metrics.

We determine the usefulness of the quantitative metrics used in the literature by comparing them to human-annotated qualitative ones to answer RQ 3. We gather seven quantitative metrics from the literature. We identify qualitative metrics from previous works and find that they often are underspecified. We contribute to the field by providing precise definitions of two that we deem the most important for our task. The two qualitative metrics, fluency and plausibility, are based on previous work on computer translation and counterfactual explanations.

To gather results of the metrics we analyze, we create an experiment described in Chapter 4. We select one counterfactual generator out of each of the derived categories. We generate and assess counterfactual explanations using quantitative and qualitative metrics. We gather ratings for the counterfactual explanations from native English speakers and central bank employees.

The sequential infilling and LLM prompting methods we analyze achieve good fluency of counterfactuals created for the FOMC dataset. However, the plausibility level of those counterfactuals is less than satisfactory. Moreover, the plausibility of the explanations is lower for generators that learn and apply domain-specific wording in their counterfactuals. Answering RQ 2, the most plausible counterfactuals for the practitioners are ones that perturb the text but use no domain-specific wording, thus having less risk of word misuse.

To answer RQ 3, we analyze the correlation between the quantitative results and the grades counterfactuals received from human annotators. We observe a strong correlation between the fluency of the explanation and the results for quantitative metrics that measure the degree of changes introduced into the counterfactuals. This correlation only holds for intra-method measurements, so it might be used to compare different methods. The smaller the change, the more fluent an explanation is. Thus, the desideratum that seems to optimize the fluency as well as plausibility is the minimality of the changes (RQ 1).

The counterfactual explanations are often classified to their target class while being rated unrealistic by the experts. This means that the counterfactual generation methods are not well suited for the task we use them for. However, these counterfactuals often achieve label flip, so they change the classifier's output label to the target class. This might point to the fact that the model trained by Shah, Paturi, and Chava [58] is not well suited for this task – either due to its relatively small size or due to the task's difficulty in the dataset.

Based on our findings, we formulate several recommendations for practitioners who aim to utilize counterfactual explanations for language models in their work. We describe these recommendations in the following section.

6.2 Recommendations for Practitioners

The primary purpose of our evaluations is to establish which of the existing methods for LM counterfactual explanations is best suited for explaining financial text LM classifiers. Through the results of our experiments and the analysis thereof, we can give recommendations for practitioners who want to use LM counterfactual generators for their tasks.

The results we present and discuss in Chapter 5 suggest that the RELITC generator is best suited for generating fluent counterfactuals. While the Polyjuice generator receives the highest scores for the plausibility of the explanations, this can be attributed to RELITC not handling the domain-specific wording properly. This could stem from the relatively small size of the data set. For a task with a larger available training set, this could mean that the RELITC counterfactual generator can be the best choice for general applications.

However, the usability of the three counterfactual generators can be task-specific. The reason for the RELITC's lower plausibility scores makes it so that the Polyjuice counterfactual generator might be preferred for applications where the domain-specific wording should not be altered.

The Polyjuice generator's advantage over the other generators is the lack of training needed for generating basic counterfactuals. Thus, the only option is to use the Polyjuice generator when there is no training set or it is too small. The generator offers satisfactory fluency, and by not altering domain-specific words, it produces plausible counterfactuals. A downside to using this generator is the relatively low flip rate of the counterfactuals, which means that it might take more time to generate a counterfactual with this method compared to the other ones.

If a practitioner is willing to post-process the counterfactuals, it could also be viable to use the PPLM generator. The generator creates a string of text based on an initial text and a conditioning class, so while its direct outputs are not usually usable as counterfactual explanations, they can act as suggestions to edit the factuals. Since the generation steps of the generator are directly steered by the classifier, its outputs can be closer to the actual target class, as shown by the lowest implausibility result in Table 5.2.

6.3 Future Work

The limitations of the presented work leave room for future work in analyzing LM counterfactual generators. Limited by the amount of expert evaluations, we cannot draw strong assumptions for aspects such as the correlation of the qualitative and quantitative metrics. Future work with higher resources might consider employing larger expert groups to evaluate text-based counterfactuals.

The lack of ground-truth counterfactuals in our analysis is an aspect of the dataset we analyze, although we also identify it as a limitation. Future work can focus on finding ways to gather factual-counterfactual pairs. This can be done through search-based methods, although for many datasets, it is not realistic to find counterfactual counterparts for large enough parts of the dataset. Expert input can also be necessary here. Creating a large financial dataset comprised of factual-counterfactual pairs can considerably help in analyzing LM counterfactual methods.

In Subsection 5.4, we discuss possible and current approaches to generating explanations using large language models like the GPT-4. We formulate a similar method of RAG-based counterfactual generators and test its performance using quantitative metrics. While the method achieved a strong label flip, it does not outperform the other counterfactual generators. To better understand the prospect those methods bring to LM explainability, they should be studied in a similar setting as ours, rigorously assessing the plausibility of their explanations. Furthermore, including the newest LLMs in generators such as PPLM and Polyjuice may bring advantages from both the high fluency of a modern LLM and the signal from the classifier in PPLM or a structured approach to counterfactual generation of Polyjuice.

6.4 Limitations

Our work is not without limitations. We select only 3 out of the multiple text counterfactual generation methods. While we attempt to consider a wide range of techniques used in the field, it is not feasible to evaluate all existing methods.

A limiting factor in using some methods is that some require additional data besides texts and labels for training purposes. The PPLM BoW attribution model requires a curated list of words for calculating the text generation direction [16]. Similarly, the work by Yang et al. [75] uses bag-of-words for an infilling task similar to the one used in RELITC. Our work analyzes the feasibility of using text counterfactual methods in real-life applications where additional data might not be available. At the same time, we acknowledge that studying those methods might bring further insights into the field. Another limitation inherent to the FOMC dataset is the lack of ground-truth counterfactuals. We considered this in designing our study since datasets acquired from real-life data usually do not contain samples with exact semantic matches in their target classes. While this consideration makes our evaluation more realistic, it does not let us evaluate the results with machine translation metrics like BLEU or include the ground-truth counterfactuals in expert evaluation. Furthermore, one cannot use some of the retrieval-based generators without factual-counterfactual pairs.

Another limitation stems from the use of a single dataset in our evaluations. While we solely consider financial text classification, the texts in this field use specific terms that might or might not be present in the pre-training data for the foundational models used in the methods we evaluate. One could gain more insight from performing similar evaluations on texts from other specialist domains, such as medicine or legal texts.

Bibliography

[1] Carlo Altavilla et al. "How do financial markets react to monetary policy signals?" In: *Research Bulletin* 73 (July 2020). URL: https://econpapers.repec.org/article/ecbecbrbu/2020_3a0073.htm (visited on 08/15/2024).

[2] Patrick Altmeyer et al. "Endogenous Macrodynamics in Algorithmic Recourse". In: *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. Raleigh, NC, USA: IEEE, Feb. 2023, pp. 418–431. ISBN: 9781665462990. DOI: 10.1109/SaTML54575.2023.00036. URL: https://ieeexplore.ieee.org/document/10136130/ (visited on 08/22/2024).

[3] Patrick Altmeyer et al. "Faithful Model Explanations through Energy-Constrained Conformal Counterfactuals". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38.10 (Mar. 2024), pp. 10829–10837. ISSN: 2374-3468. DOI: 10.1609/aaai.v38i10.28956. URL: https://ojs.aaai.org/index.php/AAAI/article/view/28956 (visited on 08/04/2024).

[4] Javier Antorán et al. *Getting a CLUE: A Method for Explaining Uncertainty Estimates*. arXiv:2006.06848 [cs, stat]. Mar. 2021. DOI: 10.48550/arXiv.2006.06848. URL: http://arxiv.org/abs/2006.06848 (visited on 08/22/2024).

[5] Andre Artelt et al. "Evaluating Robustness of Counterfactual Explanations". In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. Orlando, FL, USA: IEEE, Dec. 2021, pp. 01–09. ISBN: 9781728190488. DOI: 10.1109/SSCI50451.2021.9660058. URL: https://ieeexplore.ieee.org/document/9660058/ (visited on 08/04/2024).

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* (Sept. 2014). URL: https://www.semanticscholar.org/paper/Neural-Machine-Translation-by-Jointly-Learning-to-Bahdanau-Cho/fa72afa9b2cbc8f0d7b05d52548906610ffbb9c5 (visited on 04/30/2024).

[7] Lorenzo Betti et al. "Relevance-based Infilling for Natural Language Counterfactuals". In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. CIKM '23. New York, NY, USA: Association for Computing Machinery, Oct. 2023, pp. 88–98. ISBN: 9798400701245. DOI: 10.1145/3583780.3615029. URL: https://doi.org/10.1145/3583780.3615029 (visited on 04/27/2024).

[8] Milan Bhan et al. "TIGTEC: Token Importance Guided TExt Counterfactuals". en. In: *Machine Learning and Knowledge Discovery in Databases: Research Track*. Ed.

by Francesco Bonchi et al. Vol. 14171. Cham: Springer Nature Switzerland, 2023, pp. 496–512. ISBN: 9783031434174 9783031434181. DOI: 10.1007/978-3-031-43418-1_30. URL: https://link.springer.com/10.1007/978-3-031-43418-1_30 (visited on 08/11/2024).

[9] Jianxin Bi. "Stock Market Prediction Based on Financial News Text Mining and Investor Sentiment Recognition". en. In: *Mathematical Problems in Engineering* 2022 (Oct. 2022). Ed. by Zaoli Yang, pp. 1–9. ISSN: 1563-5147, 1024-123X. DOI: 10.1155/2022/2427389. URL: https://www.hindawi.com/journals/mpe/2022/2427389/ (visited on 08/21/2024).

[10] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[11] Christopher M. Bishop. *Pattern recognition and machine learning.* Information science and statistics. New York: Springer, 2006. ISBN: 9780387310732.

[12] Stanley F Chen, Douglas Beeferman, and Roni Rosenfeld. "Evaluation Metrics For Language Models". In: (2008), 81295 Bytes. DOI: 10.1184/R1/6605324.V1. URL: https://kilthub.cmu.edu/articles/Evaluation_Metrics_For_Language_Models/6605324/1 (visited on 08/11/2024).

[13] Yanda Chen et al. "Do Models Explain Themselves? Counterfactual Simulatability of Natural Language Explanations". en. In: (Oct. 2023). URL: https://openreview.net/forum?id=VvAiCXwPvD (visited on 08/23/2024).

[14] Zhiyu Zoey Chen et al. *A Survey on Large Language Models for Critical Societal Domains: Finance, Healthcare, and Law.* arXiv:2405.01769 [cs]. May 2024. DOI: 10.48550/arXiv.2405.01769. URL: http://arxiv.org/abs/2405.01769 (visited on 08/19/2024).

[15] Anna Cieslak and Andreas Schrimpf. "Non-monetary news in central bank communication". en. In: *Journal of International Economics* 118 (May 2019), pp. 293–315. ISSN: 00221996. DOI: 10.1016/j.jinteco.2019.01.012. URL: https://linkinghub.elsevier.com/retrieve/pii/S002219961830268X (visited on 08/15/2024).

[16] Sumanth Dathathri et al. "Plug and Play Language Models: A Simple Approach to Controlled Text Generation". In: *ArXiv* (Sept. 2019). URL: https://www.semanticscholar.org/paper/Plug-and-Play-Language-Models%3A-A-Simple-Approach-to-Dathathri-Madotto/e04a80263d252a3d8a382ba37a249b9345620570 (visited on 04/27/2024).

[17] Delft High Performance Computing Centre DHPC. *DelftBlue Supercomputer Phase 2.* 2024. URL: https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2.

[18] Pierre A. Delice et al. "The Economic Value of Words: an Evaluation of News for Economic Analysis". In: *2024 IEEE Latin American Electron Devices Conference (LAEDC).* Guatemala City, Guatemala: IEEE, May 2024, pp. 1–4. ISBN: 9798350361292. DOI: 10.1109/LAEDC61552.2024.10555884. URL: https://ieeexplore.ieee.org/document/10555884/ (visited on 08/21/2024).

[19] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Lin-

guistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423 (visited on 05/06/2024).

[20] Tanay Dixit et al. "CORE: A Retrieve-then-Edit Framework for Counterfactual Data Generation". In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 2964–2984. DOI: 10.18653/v1/2022.findings-emnlp.216. URL: https://aclanthology.org/2022.findings-emnlp.216 (visited on 04/27/2024).

[21] Kelvin Du et al. "Financial Sentiment Analysis: Techniques and Applications". en. In: *ACM Computing Surveys* 56.9 (Oct. 2024), pp. 1–42. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3649451. URL: https://dl.acm.org/doi/10.1145/3649451 (visited on 08/15/2024).

[22] Xiaoli Fern and Quintin Pope. "Text Counterfactuals via Latent Optimization and Shapley-Guided Search". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens et al. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 5578–5593. DOI: 10.18653/v1/2021.emnlp-main.452. URL: https://aclanthology.org/2021.emnlp-main.452 (visited on 04/27/2024).

[23] João Fonseca et al. "Setting the Right Expectations: Algorithmic Recourse Over Time". In: *Proceedings of the 3rd ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*. EAAMO '23. New York, NY, USA: Association for Computing Machinery, Oct. 2023, pp. 1–11. ISBN: 9798400703812. DOI: 10.1145/3617694.3623251. URL: https://dl.acm.org/doi/10.1145/3617694.3623251 (visited on 04/24/2024).

[24] Oana Frunza and Morgan Stanley. "Information Extraction from Federal Open Market Committee Statements". In: *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*. Ed. by Dr Mahmoud El-Haj et al. Barcelona, Spain (Online): COLING, Dec. 2020, pp. 195–203. URL: https://aclanthology.org/2020.fnp-1.32 (visited on 08/22/2024).

[25] Yair Gat et al. *Faithful Explanations of Black-box NLP Models Using LLM-generated Counterfactuals*. arXiv:2310.00603 [cs]. Nov. 2023. DOI: 10.48550/arXiv.2310.00603. URL: http://arxiv.org/abs/2310.00603 (visited on 04/27/2024).

[26] Stefan Gebauer, Thomas McGregor, and Julian Schumacher. "How central bank communication affects the economy". en. In: (July 2024). URL: https://www.ecb.europa.eu/press/blog/date/2024/html/ecb.blog240731~61a58b1e4a.en.html (visited on 08/15/2024).

[27] Daniel Gilo and Shaul Markovitch. "A General Search-Based Framework for Generating Textual Counterfactual Explanations". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38.16 (Mar. 2024), pp. 18073–18081. ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v38i16.29764. URL: https://ojs.aaai.org/index.php/AAAI/article/view/29764 (visited on 04/27/2024).

[28] Rishin Haldar and Debajyoti Mukhopadhyay. *Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach*. arXiv:1101.1232 [cs, math]. Jan. 2011. DOI: 10.48550/arXiv.1101.1232. URL: http://arxiv.org/abs/1101.1232 (visited on 08/10/2024).

[29] Anne Lundgaard Hansen and Sophia Kazinnik. "Can ChatGPT Decipher Fedspeak?" en. In: *SSRN Electronic Journal* (2023). ISSN: 1556-5068. DOI: `10.2139/ssrn.4399406`. URL: `https://www.ssrn.com/abstract=4399406` (visited on 08/22/2024).

[30] Tim Henderson. *Zhang-Shasha: Tree edit distance in Python — Zhang-Shasha v1.2.0.* URL: `https://zhang-shasha.readthedocs.io/` (visited on 08/10/2024).

[31] Eduard Hovy, Margaret King, and Andrei Popescu-Belis. "Principles of Context-Based Machine Translation Evaluation". en. In: *Machine Translation* 17.1 (Mar. 2002), pp. 43–75. ISSN: 1573-0573. DOI: `10.1023/A:1025510524115`. URL: `https://doi.org/10.1023/A:1025510524115` (visited on 04/29/2024).

[32] Marek Jarociński and Peter Karadi. "Deconstructing Monetary Policy Surprises—The Role of Information Shocks". en. In: *American Economic Journal: Macroeconomics* 12.2 (Apr. 2020), pp. 1–43. ISSN: 1945-7707. DOI: `10.1257/mac.20180090`. URL: `https://www.aeaweb.org/articles?id=10.1257/mac.20180090` (visited on 08/15/2024).

[33] F. Jelinek et al. "Perplexity—a measure of the difficulty of speech recognition tasks". en. In: *The Journal of the Acoustical Society of America* 62.S1 (Dec. 1977), S63–S63. ISSN: 0001-4966, 1520-8524. DOI: `10.1121/1.2016299`. URL: `https://pubs.aip.org/jasa/article/62/S1/S63/642598/Perplexity-a-measure-of-the-difficulty-of-speech` (visited on 08/11/2024).

[34] Shalmali Joshi et al. *Towards Realistic Individual Recourse and Actionable Explanations in Black-Box Decision Making Systems.* arXiv:1907.09615 [cs, stat]. July 2019. DOI: `10.48550/arXiv.1907.09615`. URL: `http://arxiv.org/abs/1907.09615` (visited on 08/22/2024).

[35] Eoin M. Kenny and Mark T. Keane. "On Generating Plausible Counterfactual and Semi-Factual Explanations for Deep Learning". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.13 (May 2021), pp. 11575–11585. ISSN: 2374-3468. DOI: `10.1609/aaai.v35i13.17377`. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/17377` (visited on 08/04/2024).

[36] Arman Khadjeh Nassirtoussi et al. "Text mining for market prediction: A systematic review". en. In: *Expert Systems with Applications* 41.16 (Nov. 2014), pp. 7653–7670. ISSN: 09574174. DOI: `10.1016/j.eswa.2014.06.009`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0957417414003455` (visited on 08/21/2024).

[37] Ben Krause et al. "GeDi: Generative Discriminator Guided Sequence Generation". In: *Findings of the Association for Computational Linguistics: EMNLP 2021.* Ed. by Marie-Francine Moens et al. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4929–4952. DOI: `10.18653/v1/2021.findings-emnlp.424`. URL: `https://aclanthology.org/2021.findings-emnlp.424` (visited on 04/27/2024).

[38] V. Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics. Doklady* (1965). URL: `https://www.semanticscholar.org/paper/Binary-codes-capable-of-correcting-deletions%2C-and-Levenshtein/b2f8876482c97e804bb50a5e2433881ae31d0cdd` (visited on 08/10/2024).

[39] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach.* arXiv:1907.11692 [cs]. July 2019. DOI: `10.48550/arXiv.1907.11692`. URL: `http://arxiv.org/abs/1907.11692` (visited on 08/21/2024).

[40] Luca Longo et al. “Explainable Artificial Intelligence (XAI) 2.0: A manifesto of open challenges and interdisciplinary research directions”. en. In: *Information Fusion* 106 (June 2024), p. 102301. ISSN: 15662535. DOI: 10.1016/j.inffus.2024.102301. URL: https://linkinghub.elsevier.com/retrieve/pii/S1566253524000794 (visited on 08/22/2024).

[41] Xiaoyi Ma and Christopher Cieri. “Corpus Support for Machine Translation at LDC”. In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC’06)*. Ed. by Nicoletta Calzolari et al. Genoa, Italy: European Language Resources Association (ELRA), May 2006. URL: http://www.lrec-conf.org/proceedings/lrec2006/pdf/754_pdf.pdf (visited on 05/15/2024).

[42] Nishtha Madaan, Diptikalyan Saha, and Srikanta Bedathur. “Counterfactual Sentence Generation with Plug-and-Play Perturbation”. In: *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. Raleigh, NC, USA: IEEE, Feb. 2023, pp. 306–315. ISBN: 9781665462990. DOI: 10.1109/SaTML54575.2023.00028. URL: https://ieeexplore.ieee.org/document/10136170/ (visited on 04/27/2024).

[43] Nishtha Madaan et al. “Generate Your Counterfactuals: Towards Controlled Counterfactual Generation for Text”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. May 2021, pp. 13516–13524. DOI: 10.1609/aaai.v35i15.17594. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17594 (visited on 04/27/2024).

[44] Puneet Mathur et al. “MONOPOLY: Financial Prediction from MONetary POLicY Conference Videos Using Multimodal Cues”. en. In: *Proceedings of the 30th ACM International Conference on Multimedia*. Lisboa Portugal: ACM, Oct. 2022, pp. 2276–2285. ISBN: 9781450392037. DOI: 10.1145/3503161.3548380. URL: https://dl.acm.org/doi/10.1145/3503161.3548380 (visited on 08/22/2024).

[45] Akira Matsui, Xiang Ren, and Emilio Ferrara. “Using Word Embedding to Reveal Monetary Policy Explanation Changes”. In: *Proceedings of the Third Workshop on Economics and Natural Language Processing*. Ed. by Udo Hahn, Veronique Hoste, and Amanda Stent. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 56–61. DOI: 10.18653/v1/2021.econlp-1.8. URL: https://aclanthology.org/2021.econlp-1.8 (visited on 08/22/2024).

[46] Clara Meister and Ryan Cotterell. *Language Model Evaluation Beyond Perplexity*. arXiv:2106.00085 [cs]. Aug. 2021. DOI: 10.48550/arXiv.2106.00085. URL: http://arxiv.org/abs/2106.00085 (visited on 08/23/2024).

[47] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. arXiv:1301.3781 [cs]. Sept. 2013. DOI: 10.48550/arXiv.1301.3781. URL: http://arxiv.org/abs/1301.3781 (visited on 08/21/2024).

[48] Ramaravind Kommiya Mothilal, Amit Sharma, and Chenhao Tan. “Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations”. In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. arXiv:1905.07697 [cs, stat]. Jan. 2020, pp. 607–617. DOI: 10.1145/3351095.3372850. URL: http://arxiv.org/abs/1905.07697 (visited on 08/22/2024).

[49] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL: probml.ai.

[50] Kishore Papineni et al. “BLEU: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL ’02. USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://dl.acm.org/doi/10.3115/1073083.1073135 (visited on 04/29/2024).

[51] Denis Peskoff et al. “GPT Deciphering Fedspeak: Quantifying Dissent Among Hawks and Doves”. In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 6529–6539. DOI: 10.18653/v1/2023.findings-emnlp.434. URL: https://aclanthology.org/2023.findings-emnlp.434 (visited on 08/22/2024).

[52] Moritz Pfeifer and Vincent P. Marohl. “CentralBankRoBERTa: A fine-tuned large language model for central bank communications”. en. In: *The Journal of Finance and Data Science* 9 (Nov. 2023), p. 100114. ISSN: 24059188. DOI: 10.1016/j.jfds.2023.100114. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405918823000302 (visited on 08/22/2024).

[53] Alec Radford et al. “Improving Language Understanding by Generative Pre-Training”. In: 2018. URL: https://www.semanticscholar.org/paper/Improving-Language-Understanding-by-Generative-Radford-Narasimhan/cd18800a0fe0b668a1cc19f2ec95b5003d0a5035 (visited on 08/21/2024).

[54] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: 2019. URL: https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe (visited on 08/11/2024).

[55] Marcel Robeer, Floris Bex, and Ad Feelders. “Generating Realistic Natural Language Counterfactuals”. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Ed. by Marie-Francine Moens et al. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3611–3625. DOI: 10.18653/v1/2021.findings-emnlp.306. URL: https://aclanthology.org/2021.findings-emnlp.306 (visited on 04/27/2024).

[56] Alexis Ross, Ana Marasović, and Matthew Peters. “Explaining NLP Models via Minimal Contrastive Editing (MiCE)”. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 3840–3852. DOI: 10.18653/v1/2021.findings-acl.336. URL: https://aclanthology.org/2021.findings-acl.336 (visited on 04/27/2024).

[57] Marek Rozkrut et al. “Quest for central bank communication: Does it pay to be “talkative”?” en. In: *European Journal of Political Economy* 23.1 (Mar. 2007), pp. 176–206. ISSN: 01762680. DOI: 10.1016/j.ejpoleco.2006.09.011. URL: https://linkinghub.elsevier.com/retrieve/pii/S0176268006000917 (visited on 08/14/2024).

[58] Agam Shah, Suvan Paturi, and Sudheer Chava. “Trillion Dollar Words: A New Financial Dataset, Task & Market Analysis”. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 6664–6679. DOI: 10.18653

/v1/2023.acl-long.368. URL: https://aclanthology.org/2023.acl-long.368 (visited on 06/29/2024).

[59] Lee A. Smales. "Classification of RBA monetary policy announcements using ChatGPT". en. In: *Finance Research Letters* 58 (Dec. 2023), p. 104514. ISSN: 15446123. DOI: 10.1016/j.frl.2023.104514. URL: https://linkinghub.elsevier.com/retrieve/pii/S1544612323008863 (visited on 08/22/2024).

[60] Timo Speith. "A Review of Taxonomies of Explainable Artificial Intelligence (XAI) Methods". en. In: *2022 ACM Conference on Fairness, Accountability, and Transparency*. Seoul Republic of Korea: ACM, June 2022, pp. 2239–2250. ISBN: 9781450393522. DOI: 10.1145/3531146.3534639. URL: https://dl.acm.org/doi/10.1145/3531146.3534639 (visited on 08/22/2024).

[61] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, Aug. 2017, pp. 3319–3328. (Visited on 08/22/2024).

[62] Paul C. Tetlock, Maytal Saar-Tsechansky, and Sofus Macskassy. "More Than Words: Quantifying Language to Measure Firms' Fundamentals". en. In: *The Journal of Finance* 63.3 (June 2008), pp. 1437–1467. ISSN: 0022-1082, 1540-6261. DOI: 10.1111/j.1540-6261.2008.01362.x. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.2008.01362.x (visited on 08/21/2024).

[63] Ellen Tobback, Stefano Nardelli, and David Martens. "Between Hawks and Doves: Measuring Central Bank Communication". en. In: *SSRN Electronic Journal* (2017). ISSN: 1556-5068. DOI: 10.2139/ssrn.2997481. URL: https://www.ssrn.com/abstract=2997481 (visited on 08/22/2024).

[64] Marcos Treviso et al. "CREST: A Joint Framework for Rationalization and Counterfactual Text Generation". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 15109–15126. DOI: 10.18653/v1/2023.acl-long.842. URL: https://aclanthology.org/2023.acl-long.842 (visited on 04/27/2024).

[65] Arnaud Van Looveren and Janis Klaise. "Interpretable Counterfactual Explanations Guided by Prototypes". en. In: *Machine Learning and Knowledge Discovery in Databases. Research Track*. Ed. by Nuria Oliver et al. Cham: Springer International Publishing, 2021, pp. 650–665. ISBN: 9783030865207. DOI: 10.1007/978-3-030-86520-7_40.

[66] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 04/30/2024).

[67] Leandro Von Werra et al. "Evaluate & Evaluation on the Hub: Better Best Practices for Data and Model Measurements". en. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Abu Dhabi, UAE: Association for Computational Linguistics, 2022, pp. 128–136. DOI: 10.18653/v1/2022.emnlp-demos.13. URL: https://aclanthology.org/2022.emnlp-demos.13 (visited on 08/11/2024).

[68] Sandra Wachter, Brent Mittelstadt, and Chris Russell. *Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR.* arXiv:1711.00399 [cs]. Mar. 2018. DOI: `10.48550/arXiv.1711.00399`. URL: `http://arxiv.org/abs/1711.00399` (visited on 04/23/2024).

[69] Yifei Wang. *Aspect-based Sentiment Analysis in Document – FOMC Meeting Minutes on Economic Projection.* arXiv:2108.04080 [cs]. Apr. 2023. DOI: `10.48550/arXiv.2108.04080`. URL: `http://arxiv.org/abs/2108.04080` (visited on 08/22/2024).

[70] John S. White, Theresa A. O'Connell, and Francis E. O'Mara. "The ARPA MT Evaluation Methodologies: Evolution, Lessons, and Future Approaches". In: *Proceedings of the First Conference of the Association for Machine Translation in the Americas.* Columbia, Maryland, USA, Oct. 1994. URL: `https://aclanthology.org/1994.amta-1.25` (visited on 04/29/2024).

[71] Tongshuang Wu et al. "Polyjuice: Generating Counterfactuals for Explaining, Evaluating, and Improving Models". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers).* Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 6707–6723. DOI: `10.18653/v1/2021.acl-long.523`. URL: `https://aclanthology.org/2021.acl-long.523` (visited on 04/27/2024).

[72] Xing Wu et al. "Mask and Infill: Applying Masked Language Model for Sentiment Transfer". en. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence.* Macao, China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 5271–5277. ISBN: 9780999241141. DOI: `10.24963/ijcai.2019/732`. URL: `https://www.ijcai.org/proceedings/2019/732` (visited on 04/27/2024).

[73] Frank Z. Xing, Erik Cambria, and Roy E. Welsch. "Natural language based financial forecasting: a survey". en. In: *Artificial Intelligence Review* 50.1 (June 2018), pp. 49–73. ISSN: 0269-2821, 1573-7462. DOI: `10.1007/s10462-017-9588-9`. URL: `http://link.springer.com/10.1007/s10462-017-9588-9` (visited on 08/21/2024).

[74] Linyi Yang et al. "Exploring the Efficacy of Automatically Generated Counterfactuals for Sentiment Analysis". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers).* Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 306–316. DOI: `10.18653/v1/2021.acl-long.26`. URL: `https://aclanthology.org/2021.acl-long.26` (visited on 04/27/2024).

[75] Linyi Yang et al. "Generating Plausible Counterfactual Explanations for Deep Transformers in Financial Text Classification". In: *Proceedings of the 28th International Conference on Computational Linguistics.* Ed. by Donia Scott, Nuria Bel, and Chengqing Zong. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 6150–6160. DOI: `10.18653/v1/2020.coling-main.541`. URL: `https://aclanthology.org/2020.coling-main.541` (visited on 04/27/2024).

[76] Kaizhong Zhang and Dennis Shasha. "Simple Fast Algorithms for the Editing Distance between Trees and Related Problems". en. In: *SIAM Journal on Computing* 18.6 (Dec. 1989), pp. 1245–1262. ISSN: 0097-5397, 1095-7111. DOI: `10.1137/0218082`. URL: `http://epubs.siam.org/doi/10.1137/0218082` (visited on 08/10/2024).

[77] Susan Zhang et al. *OPT: Open Pre-trained Transformer Language Models*. arXiv:2205.01068 [cs]. June 2022. DOI: `10.48550/arXiv.2205.01068`. URL: `http://arxiv.org/abs/2205.01068` (visited on 08/23/2024).

[78] Yaoming Zhu et al. “Texygen: A Benchmarking Platform for Text Generation Models”. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR ’18. New York, NY, USA: Association for Computing Machinery, June 2018, pp. 1097–1100. ISBN: 9781450356572. DOI: `10.1145/3209978.3210080`. URL: `https://doi.org/10.1145/3209978.3210080` (visited on 04/29/2024).

Appendices

Appendix A

Informed Consent Form

You are being invited to participate in a Master's thesis research study titled Evaluating Language Model Explanations in Specialist Fields. This study is being done by Karol Dobiczek under supervision of Cynthia C.S. Liem and Patrick Altmeyer from the TU Delft.

The purpose of this research study is to assess the usability of modern language model explainability tools in generating texts in specialist fields, such as finance. This study will take you approximately 15 minutes to complete. The data will be used for evaluating a counterfactual explanation method. We will be asking you to rate pieces of text on a number of criteria using a 1 to 5 scale, and describe your reasoning in open questions.

As with any online activity the risk of a breach is always possible. To the best of our ability your answers in this study will remain confidential. We will minimize any risks by only collecting your personal information for the purpose of verification of the identity of the respondents. In our research we will pseudonymize your identity and solely use the answers to the questions relating to text assessment. The survey data will be stored on a Project Storage drive at TU Delft and all personal information will be destroyed after the end of the thesis project.

Your participation in this study is entirely voluntary and you can withdraw at any time. You are free to omit any questions.

Contact details for the corresponding researcher:
Karol Dobiczek
k.t.dobiczek@student.tudelft.nl
Master's Student, Multimedia Computing Group

By submitting a response to this survey you agree to this Opening Statement and to your response being used for the research described above, and for your de-identified answers to be included in the final data set that will be publicly available when the research is published. I understand that once my response has been submitted my data will have been processed in such a way that it is no longer possible for it to be withdrawn.

Appendix B

Survey Topic Introduction

Counterfactual Explanations are a form of explainable AI aiming to explain a classification made by a Machine Learning model by proposing an alternative to the original input. Imagine you write a text that you intend to be perceived as positive, but a sentiment analysis Language Model doesn't find it quite convincing. Through a counterfactual explanation, we can generate a text which could better reflect the intended tone.

Your task:

We will present you with several counterfactual sentences generated via different means. On each page, we will show you an original (factual) sentence and three variants of counterfactuals. We will ask you to **grade the sentences** you see using the following criteria:

Fluency: A fluent segment is one that is grammatically well-formed; contains correct spellings; adheres to the common use of terms, titles and names; contains properly capitalized letters; and is intuitively acceptable. Unfinished sentences also impact the fluency of a segment.

Please rate the texts using this definition of fluency. A text should receive a score of:

- 5/5 if it follows this definition completely.
- 3/5 if there are several mistakes but the text still is interpretable.
- 1/5 if it is not fluent or grammatically correct English.

`For expert evaluation only:`

Plausibility: A plausible counterfactual segment adheres well to samples seen in the real data distribution, and the target sentiment of the target sentence class. The changes made to the factual, considering the meaning and context of the edited words, should also fit the target domain.

Please rate the texts using this definition of plausibility. A text should receive a score of:

- 5/5 if it follows this definition completely.

- 3/5 if there are several mistakes but the text reflects the right sentiment.
- 1/5 if the changes are nonsensical.

These criteria will also appear at the end of each page.

In an **open question**, we will ask you to describe what qualities that you might look for in a text like this are missing. Your comment can refer to the semantics of the sentence, its structure, or its contents. If you do not have any comments you can also leave the answer empty.

The order of the methods used for each question will be randomized.

Appendix C

Additional Quantitative Results

C.1 Alternative Models for Perplexity Calculation

	facebook/opt-125m		gpt2		lxyuan/distilgpt2-finetuned-finance	
	Perplexity	Perpl. ratio	Perplexity	Perpl. ratio	Perplexity	Perpl. ratio
Polyjuice	107.06 (291.9)	1.90 (7.9)	90.98 (172.1)	1.80 (4.6)	104.06 (150.3)	1.62 (3.84)
PPLM	**36.07** (15.9)	**0.68** (0.4)	**43.90** (23.5)	**0.78** (0.5)	**43.89** (23.5)	**0.69** (0.4)
RELITC	108.86 (153.8)	1.52 (0.8)	100.95 (125.2)	1.67 (1.2)	111.99 (142.0)	1.52 (1.0)

Table C.1: Comparison of perplexity-based metrics computed using three language models. The base GPT-2, an Open Pretrained Transformer (OPT) [77] opt-125m (`https://huggingface.co/facebook/opt-125m`), and a GPT-2 model fine-tuned on four financial datasets (`https://huggingface.co/lxyuan/distilgpt2-finetuned-finance`).

C.2 Quantitative Results of Successful Counterfactuals

	Perplexity	Perp. ratio	Edit dist.	Tree dist.	Embedding dist.	Impl.
Polyjuice	99.64 (227.0)	1.91 (4.6)	0.36 (0.3)	22.10 (21.7)	**20.35** (4.1)	**29.06** (3.4)
PPLM	**36.64** (16.2)	**0.77** (0.4)	0.76 (0.6)	36.25 (6.7)	20.69 (3.7)	29.56 (2.9)
RELITC	104.04 (130.2)	1.68 (1.3)	**0.12** (0.1)	**9.90** (13.2)	21.84 (3.8)	33.35 (3.5)

Table C.2: Quantitative results computer over results containing only successful counterfactuals.

Appendix D

Pseudo-RAG Generator

The size of new LLMs, such as the GPT-4 or Mistral-7B, prevents those models from being part of counterfactual generators, like the GPT-2 in PPLM. Due to that, the quality of the contextual generators using older models might be lower compared to that possible with the use of new LLMs. The newer generation of language models has been shown to perform even better than their processors on zero short tasks, so one might assume that their accuracy and their performance for a counterfactual generation task might also be good. We thus performed an experiment using the GPT-4o large language model to create a counterfactual generator, and we tested it on the FOMC task.

In designing our proof of concept method, we take inspiration from the retrieval-augmented generation technique. In Retrieval-Augmented Generation, a large language model is supplied with a number of texts or documents that the user's query relates to; the model is then tasked with answering the user's query using the contents of the documents. While there are several approaches to counterfactual generation using RAG, they all rely on data sets that contain factual-counterfactual pairs, pairs that the FOMC dataset, among many others, lacks. This is a severe limitation because the generators can only be applied to a handful of specific datasets. In view of this limitation, we decide to supply the LLM with several examples of factual sentences from both the factual class and the target class creating a pseudo-RAG generator. We then ask the model to create a new counterfactual that could be classified to the target class by making as few changes to the original sentence as possible.

Table D.1 shows the results of text counterfactual generation using our pseudo-RAG method. As with the previous experiments, we designed the experiment to use a reasonable number of generation attempts, generating five counterfactuals per factual text. Even with the small amount of counterfactuals generated, the method achieves the highest flip rate score of 0.88. While the perplexity results for PPLM are still better than this proof of concept, we get the lowest perplexity, the second lowest perplexity out of the four generators. However, the results of the other metrics are comparable with the other methods. On the other hand, the quality of the generated sentences is seemingly the best

A classification Machine Learning model classifies texts into three classes: DOVISH, HAWKISH and NEUTRAL. Your task is to transform a QUERY sentence that was classified as `{label}` into a COUNTERFACTUAL that should be classified as `{target}`. You can replace, remove or add words, but you should keep the amount of changes to minimum, only performing up to 5 changes. You can use the EXAMPLE `{factual label}` and EXAMPLE `{target label}` sentences as examples how sentences belonging to those classes might look like. You should generate only one COUNTERFACTUAL sentence.

EXAMPLE `{factual label}`:
`{factual class examples}`

EXAMPLE `{target label}`:
`{target class examples}`

`{factual label}` QUERY: `{factual}`

`{target label}` COUNTERFACTUAL:

Figure D.1: Prompt of the proof of concept GPT-3.5 generator.

out of all generators. This is probably due to the complexity of the model and the higher quality of the outputs compared to the other models. A notable result is the implausibility metric, where this model receives the highest score, meaning that the embeddings of the counterfactuals generated by this model are furthest away from the factuals in our data set.

Similarly to Polyjuice, our proof of concept method has no information about the classifier. However, similarly to PPLM, it has no restrictions with regard to the amount of tokens generated, so the changes it generates are not controlled, which can cause the counterfactuals to stray away from the factual sentences. The poor results of the implausibility metric, combined with the high accuracy and seemingly high quality of the counterfactuals, point us to believe that involving the classifier and generating counterfactuals is important, especially for classification tasks. While this model can be useful for generating new data sets or new training sets, it is not likely to be used to generate useful explanations for classification tasks.

D.1 Results

	Perplexity	Perpl. ratio	Edit dist.	Tree dist.	Embedding dist.	Impl.	Succ. rate
Pseudo-RAG	74.00 (38.8)	1.37 (0.5)	0.29 (0.1)	19.40 (11.5)	24.86 (4.0)	32.39 (2.9)	**0.88**
Polyjuice	86.49 (79.9)	1.58 (1.3)	0.26 (0.3)	17.36 (15.3)	**24.78** (3.5)	**31.56** (2.7)	0.36
PPLM	**37.11** (15.2)	**0.76** (0.4)	0.56 (0.2)	37.48 (7.3)	24.97 (4.4)	32.09 (4.5)	0.52
RELITC	86.72 (71.6)	1.54 (1.0)	**0.13** (0.1)	**11.00** (7.0)	25.83 (3.7)	32.18 (3.1)	0.80

Table D.1: Results for the quantitative metrics including the Pseudo-RAG method. Averaged over 25 factual-counterfactual rows.