Submitted to the Annals of Applied Statistics

INTEGRATE ATTENTION-BASED NEURAL NETWORKS WITH TEXT REGRESSIONS IN SENTIMENT INFERENCE

BY LÉON YUAN[1,a]

[1]*Department of Statistical Science, Southern Methodist University,* [a]*yuanli@smu.edu*

This paper serves to summarize three papers that are related to sentiment inference and text regression. The first paper surveys relevant statistical text analysis methods and their applications. The second paper concerns a new proposed attention-based multiple-instance classification model (AMIC). The third paper augments data with counterfactual examples to improve the models' generality. I discussed each paper in turn and synthesized them together. Finally, I reproduced some computational results of the AMIC method.

1. Introduction. The main focus of the research is *Text Sentiment Analysis*. The origin of *Text Sentiment Analysis* can date back to 2002 when Turney (2002) first proposed a simple unsupervised learning algorithm that could classify reviews into "Thumbs Up" and "Thumb Down". Turney (2002) estimated the semantic orientation of each phrase and calculated the average of all semantic orientations of each review to classify. This high-level idea resembles the first paper (Ph.D. Thesis) Yang (2023) that I discussed later a lot. This research benefits both academics and applications. Regarding academics, *Text Sentiment Analysis* drives researchers to propose new classification algorithms or improve existing methods. This research can be extended to topic modeling and clustering of journal articles or books. Topic modeling learns the underlying patterns between unspecified topics and words in the corpus in an unsupervised way. Probabilistic Latent Semantic Analysis and Latent Dirichlet Allocation are two popular methods for topic modeling. Clustering academic articles and books makes it easy for researchers to find relevant resources for various research. This process encourages researchers in different domains, such as statistics, computer science, and linguistics, to participate in interdisciplinary collaboration. Regarding applications, business firms need to classify numerous reviews, reports, emails, and chat transcripts to make informed decisions and gain profits by applying *Text Sentiment Analysis*. For instance, a product company can track its performance by collecting online customer reviews to gain insight. The challenge is that their products can produce hundreds of customer reviews each day. The product managers cannot read them all. Then, this company needs an automatic tool to monitor its product performance. Sentiment Analysis is the tool that most companies

*Ph.D. Qualifying Exam 2024

Keywords and phrases: tranformers, text regression.

apply to their business. It automatically detects how many positive, negative, and neutral customer reviews and reports such a summary to the product manager. This product manager then provides feedback to the product engineering team to improve their designs. As a result, this company improves their customer service and their reputation. In academics, researchers, especially computer scientists, developed deep neural networks McCulloch and Pitts (1943) such as convolutional neural networks Krizhevsky, Sutskever and Hinton (2012), transformers Wolf et al. (2019), "Bidirectional Encoder Representation from transformers" (BERT) Devlin et al. (2018), and so on to make natural language understanding and inference. MacCartney (2014) illustrated the high-level terminologies and their relations in the natural language easily by the following diagram 1.

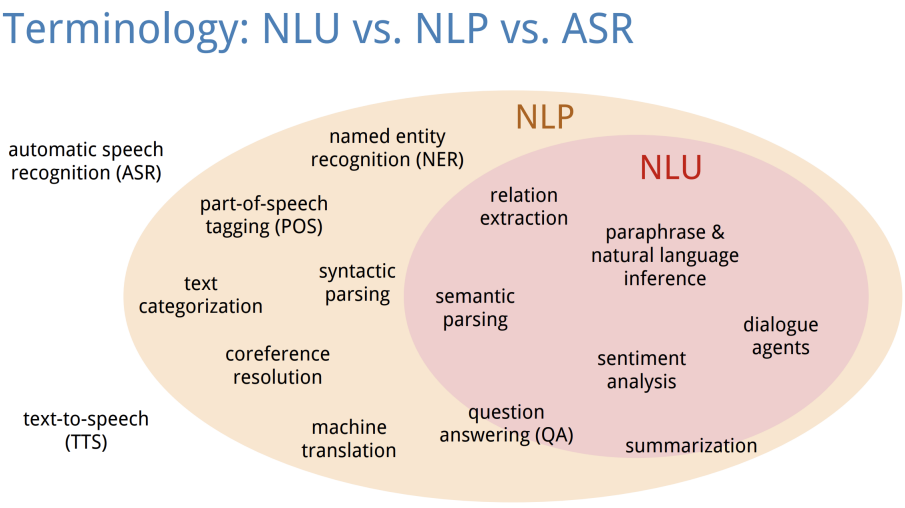


FIG 1. *Terminologies their relations*

In the second section, I discussed three core papers in turn. I first discussed the paper "**Text as Data**" because Gentzkow, Kelly and Taddy (2019) introduced why we used text as inputs to economic and social science and how to represent text in a suitable format for scientific analysis. Next, I discussed the paper "**Interpretable sentiment analysis using the attention-based multiple instance classification model**". Yang (2023) proposed a word-level attention-based neural network model for sentiment analysis. Finally, I discussed the paper "**learning the difference that makes a difference with counterfactually-augmented data**". Kaushik, Hovy and Lipton (2019) augmented text data with counterfactual examples to improve the versatility of models. This paper Kaushik, Hovy and Lipton (2019) can serve as a future direction of Yang (2023) on wine reviews for sentiment analysis.

2. Analysis of Core Papers.

2.1. *The first paper.* This first paper is called "**Text as Data**" Gentzkow, Kelly and Taddy (2019) written by Matthew Gentzkow, Bryan Kelly, and Matt Taddy. This paper has three main sections: 1. represent text as data, 2. statistical methods, and 3. applications. Human beings started recording and saving digital data dating back to 1940. However, most of them were numeric. Starting from 1980 and beyond, as more people could own personal computers and the internet started to take off, a large amount of digital data has been collected, shared, and saved. Such digital data includes news, articles, books, podcasts, reports, videos, and social media. All kinds of data contain text. In the 1970s, the earliest natural language processing was regular expression created in Unix Operating System. Starting from the 1980s, as computational linguistics was developed, researchers started building machines that could understand texts mimicking humans' brains and interact with humans by parsing the syntax and representing semantic. However, for the past 30 years, researchers in computational linguistics have shifted their perspective of nonprobabilistic syntactic semantics to quantitative probabilistic words in a stochastic process. This later perspective is named **Statistical NLP**, the mainstream focus to date. One of the most influential Statistical NLP books is called **"Foundations of Statistical Natural Language Processing"** written by Manning and Schütze (1999).

2.1.1. *Represent Text as Data.* In the last paragraph, I discussed plenty of digital texts, and **Statistical NLP** is the most effective way to deal with them. However, all traditional statistical methods need numeric inputs. Then, a fundamental question is how we convert texts into traditional data formats. The most straightforward way of representing text in data is to count the appearance of unique words of a corpus in each document. However, in practice, many documents form a corpus. As a result, the dimension of all unique words across all documents is extremely high, making the statistical analysis infeasible. Researchers came up with some ways to reduce the dimensions. The first way is to remove "**Stop Words**" such as "*a, an, the, for, about, it, and*" and so on. The reason for removing **Stop Words** is that these words are usually grammatically functional in sentences instead of expressing meanings. Since 1988 Doyle published a list of common stop words for over 40 languages. SPARCK JONES (1972) proposed a very widely used approach, TF-IDF, to highlight important characteristic words in documents while reducing the commonly used words by assigning different weights to each word. The second way is to filter out words assigned weights by **TF-IDF** lower than a predefined threshold. Hiemstra (2009) described an alternative method of representing text as data, **N-Gram**, which uses a phrase of length n as the minimal unit instead of one word. Such a method considers term dependencies and orders. For example, a bigram model counts "*What's up*" as one phrase consistent with human meaning. However, a unigram model counts it as "*What's*" and "*up*", which breaks the original meaning. But dimensions of **N-Gram** model explode as **N** increases. As a result, researchers usually first try unigram representation and then evaluate if it is worth trying a 2-gram representation.

All the methods above represent texts as counts. However, counting tokens do not represent all the information conveyed in the text because they ignore word context dependency and word order position. So, researchers have been proposing new ways to represent text as data. Mikolov et al. (2013) proposed **Word2Vec** method including **Skip-Gram** and **Negative Sampling Algorithm** to train vector representations of words based on their contexts across the whole corpus. The most impressive aspect of this method is that vector representations of words with similar meanings are located in a proximity area of the whole vector space. This means a simple vector operation produces a meaningful result. For example,

$$vector(\text{"}Beijing\text{"}) - vector(\text{"}China\text{"}) \approx vector(\text{"}Tokyo\text{"}) - vector(\text{"}Japan\text{"})$$

Without informing models what a country is and capital is, **Word2Vec** can group words of countries and learn the implicit relations between countries and capitals. Such implicit learning happens to other embeddings. Mikolov et al. (2013) illustrated their results by the following graph 2, and they built a tool for computing continuous distributed representations of words at Word2Vec.

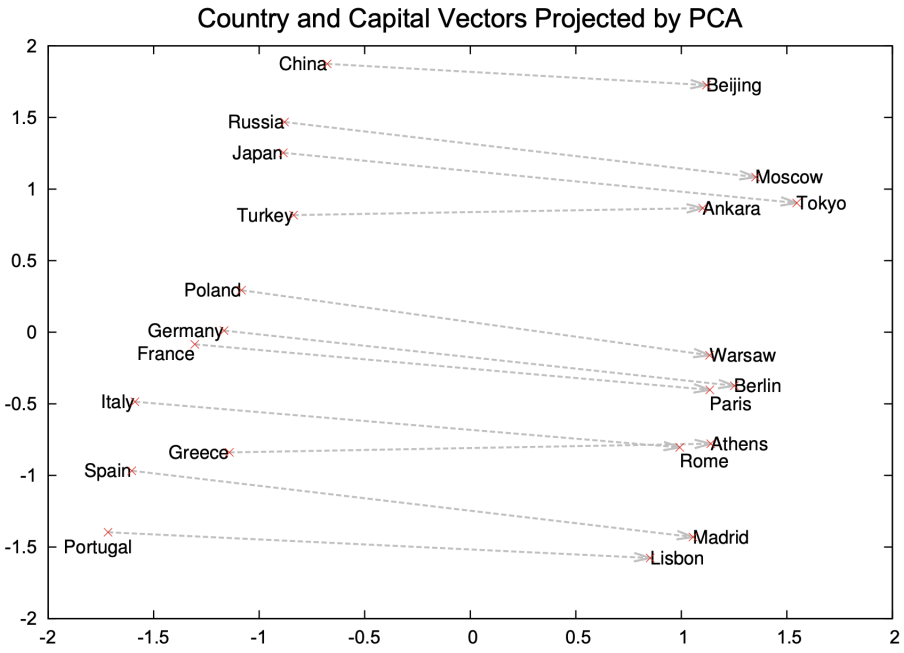


FIG 2. *Word to Vector Embedding*

Based on the work by Mikolov et al. (2013), Pennington, Socher and Manning (2014) proposed a new method of representing words as embeddings, **GloVe** (Global Vector). This new method used a word-word co-occurrence matrix to capture global and local contexts. This word-word co-occurrence matrix measures how frequently one word appears near another word across the whole corpus. Pennington, Socher and Manning (2014) illustrated the conditional probability of a word-word co-occurrence matrix with the

following figure 3. The equation "$\frac{P(k|ice)}{P(k|steam)} = 8.9$" from this figure 3 indicates that *solid* co-occurs more frequently with *ice* than *steam*. This example is consistent with human logic. As a result, this new method outperformed the **Word2Vec** method on word analogy tasks and maintained the linear relation property of word embeddings in the vector space. Pennington, Socher and Manning (2014) built a website and published their project codes for GloVec.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

FIG 3. *Conditional Probability of a Word-Word Co-occurrence Matrix*

2.1.2. *Analysis Methods.* I discussed many methods of representing text as data in the last section. Then, once we have text data, researchers propose many methods to analyze them. There are two groups of methods. The first group is to analyze relationships between token counts and attributes of interest with traditional statistical methods. This group provides interpretations and is ready for causal inference. The second group focuses on predicting attributes of interest with word embeddings and neural networks. This latter group usually has a higher prediction accuracy than the first group. However, it lacks some interpretations and is hard to reason about inference.

The first group of methods can be divided into two subgroups. The first subgroup of methods is **dictionary-based** methods. For instance, in the case of sentiment analysis, this method classifies documents by counting words in a predefined list to indicate positive or negative tones. Reid, McKenny and Short (2023) provided some practices about dictionary-based methods. The second subgroup of methods is **text regression**. **Text regression** methods regress attributes of interest on document-term count matrix with traditional statistical methods. Attributes of interest can be sentiment or topics. A simple document-term count matrix is illustrated by the table 1. Linear regression is sometimes applied to text regression because

	I	love	hate	you
Document 1	1	0	1	1
Document 2	1	1	0	1

TABLE 1
Document-Term Matrix

it is fast, intuitive, interpretative, and has many statistical properties. However, due to the extremely high dimensions and the sparsity of the document-term matrix, the ordinary linear regression performs poorly

on text regression. So, **Penalized Linear Regression** is applied. Hoerl and Kennard (1970) introduced biases into estimations of coefficients in multivariate linear regression to reduce the variance and improve model stability. This technique successfully resolves multicollinearity issues. Later Tibshirani (1996) proposed a new interpretable method, **Lasso**, that can select variables and make models stable like the ridge regression by a condition that the sum of absolute coefficients needs to be less than a preset constant. This is a cornerstone in the field of statistical methods because **Lasso** can also be extended to generalized regression models and tree-based models. Combining the ridge regression and lasso, Zou and Hastie (2005) proposed a new regularization and variable selection method named **Elastic Net** that often outperforms lasso, especially when the number of variables is larger than the number of observations. This applies to the document-term matrix, in which the number of unique tokens (columns) is usually larger than the number of documents (rows). Later, Candès, Wakin and Boyd (2008) proposed the **log penalty**, which can be used as a new regularization method. Gentzkow, Kelly and Taddy (2019) drew a picture 4 to compare these four regularization functions side by side. Elastic Net is the most widely used penalized term in linear regression. Another common technique of reducing high dimensions is **principal component analysis**

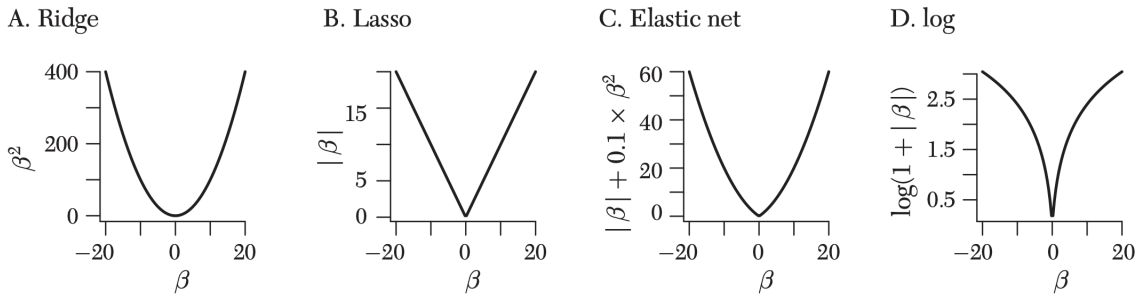


Figure 1

FIG 4. *Different regularization functions*

(PCA) proposed by HOTELLING (1933), which reduces the number of predictors as many as we want while keeping as much information as the original data has. This technique soon became the most popular dimension reduction method in statistics and data science. However, PCA does not consider how the information about attributes affects the structure of all predictors. So, a Swedish statistician Wold (2004) proposed **Partial Least Regression** to reduce the dimensions by taking advantage of relations between response variables and covariates. Taddy (2013) proposed **Multinomial Inverse Regression** method that reduces the high-dimensional document-term matrix to have a rich information representation in sentiment analysis. He also proposed the **gamma-lasso** algorithm to get a stable and effective estimation of his new method. The advantage of his method is that it can give an interpretable word loading, which quantifies

the degree of sentiment at the word level, and his method is faster to run on a large corpus with limited resources than deep-learning-based methods that I discussed later. Yang (2023) compared Taddy (2013)'s method with his new method in the second paper.

All the above-mentioned methods are linear. Although linear models are simple and easy to interpret, many data-generating processes don't follow linearity. Nonlinear text regression is the tool to tackle them. There are three major nonlinear methods used in text analysis – **generalized linear models**, **support vector machines**, and **tree-based models**. Nelder and Wedderburn (1972) unified a wide range of statistical models with a common framework, **generalized linear models**, and provided a unified procedure to fit normal and non-normal data. Their work is a groundbreaking foundation for the whole statistical field. Boser, Guyon and Vapnik (1992) proposed **Support Vector Machine** (SVM), which can transform text representations into a higher dimension space where a hyperplane that can divide different classes of texts exists. Breiman (2001) proposed a new tree-based method called **Random Forest**, which can be used for classification and regression tasks. Later, based on random forest, Chen and Guestrin (2016) proposed new algorithms that enable **XGBoost** to compute large-scale data tasks with limited resources. We mostly consider Support Vector Machine, Random Forest, and XGBoost as three major Machine Learning algorithms. They are applied not only in text analysis but also in other fields by data scientists and statisticians every day.

Next, I discuss the second group of methods based on neural networks. Bengio, Ducharme and Vincent (2000) first applied neural networks to learn distributed representations of words and made language models a significant step into deep learning-based methods. A theoretical foundation supports why deep-learning-based methods work the best among language models. Cybenko (1989) proposed and proved that a neural network can uniformly approximate any continuous function. This significant desired property is named **Universal Approximator**. Kim (2014) built **convolutional neural networks** (CNN) to a sentence-level classification task with pre-trained 300-dimension word embeddings from **Word2Vec** method as inputs proposed by Mikolov et al. (2013). His CNN models performed very well on some benchmark datasets, and he remarked on the effectiveness of CNN models in sentiment analysis and question classification. Kim (2014) described his simple CNN architecture in the following figure 5. Hochreiter and Schmidhuber (1997) proposed a gradient-based method, **Long Short-Time Memory** (LSTM), that learns to bridge long-time lag tasks. Later, Liu et al. (2016) applied a word-level bidirectional LSTM (biLSTM) to the Stanford Natural Language Inference Corpus and showed it performed well. Next, Vaswani et al. (2017) published a paper called **Attention Is All You Need**, which revolutionizes the whole field of natural language understanding. It is the foundation of modern language modeling, **Transformer-based** architecture. The biggest advantage of it is that it captures all words in one sentence at once and pays attention to keywords with the multi-head self-attention mechanism. Previous language models process texts word by word sequentially. As a result, transformer-based models can capture longer dependency contexts. Moreover, it can be easily extended to

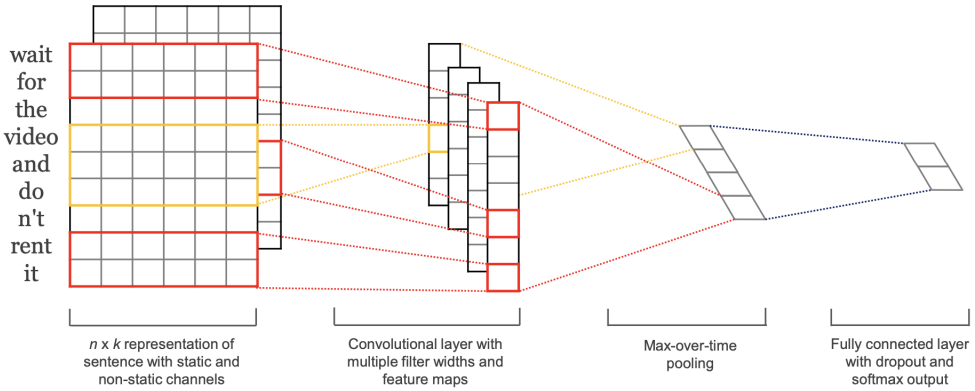


FIG 5. *A simple two-channel CNN architecture in NLP*

other language tasks and performs well. Vaswani et al. (2017) explained transformer model architecture by the following diagram 6. Combing Liu et al. (2016)'s bidirectional LSTM and Vaswani et al. (2017)'s

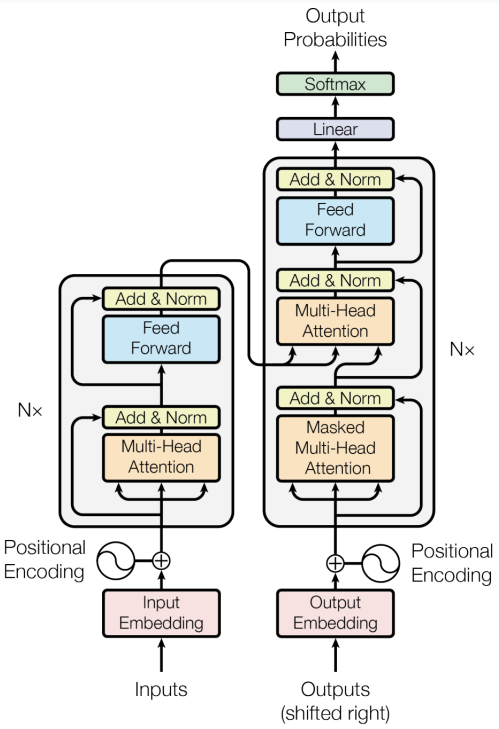


FIG 6. *The Transformer Model Architexture*

transformer model, Devlin et al. (2018) proposed a new language representation called **"Bidirectional Encoder Representations from Transformers"**, BERT, which achieved the state-of-art performance on

some common dataset. People can add one task-specified layer on top of a pre-trained BERT model. Then, people can fine-tune it for a needed task. Large Language Models are based on BERT, including ChatGPT.

2.1.3. *Applications.* Text Analysis can be applied to a wide range of fields. Researchers need it to infer the authorship of debuted articles, books, or text. Economists use it to predict stock prices with news articles. They also use it to measure economic policy uncertainty and its effects. Social scientists use it to observe and interpret news outlets' slants and how they affect or manipulate public opinion. Text analysis is also widely used in law and politics. For instance, integrating congress speeches with time series sheds light on how politics or ideology in a country change as time goes by.

2.2. *The second paper.* The natural language understanding task in the second paper is sentiment analysis. Zhang, Wang and Liu (2018) provided a comprehensive survey of deep learning methods in sentiment analysis. Most deep-learning-based methods are challenging to interpret and make further inferences. Although they achieve state-of-the-art performance in natural language understanding, they are criticized as **Black Box** to humans. People know they work very well, but people don't know why they work so well. A Ph.D. Murdoch (2019) advised by one of the most Distinguished Statistical Professor Bin Yu from UC Berkeley wrote his Ph.D. thesis about interpreting deep learning for natural language processing. Yang (2023) proposed a word-level attention-based multiple instance classification model (AMIC). The most significant advantage of the AMIC method is that it provides a word-level context-based sentiment score. This makes a neural network-based model accessible and interpretable. Regarding sentiment analysis, some other researchers have made some progress in interpreting neural networks.

Next, I dived deeper into the AMIC methodology proposed by Yang (2023). His method aims to do sentiment analysis. There are three main components in AMIC. The first component represents texts as word embeddings with **GloVec-300-Wiki** proposed by Pennington, Socher and Manning (2014). The second component is the Bayesian multiple-instance classification (MIC), proposed by Xiong (2021). MIC consists of two regression models. The first model is selecting words that express sentiment in one document. The second model uses selected words to make sentiment predictions for one document. The third component uses a self-attention mechanism to capture local and global contexts. It helps the second component select words based on word meaning and contexts.

The second component, MIC, is a core part of the model. Yang (2023) drew a figure 7 to show his model architecture. The index i in figure 7 represents the i^{th} document in a corpus. The index j in figure 7 represents the j^{th} word. Then x_{ij} denotes j^{th} word context-independent embedding in i^{th} document. y_i denotes the true i^{th} document sentiment label. Both x_{ij} and y_i are observed. Table 2 illustrates this data structure. MIC consists of two regression models, which can be represented in two layers.

The right blue dashed square box in figure 7 represents layer one, which is trained to get indicator values, δ_{ij}. δ_{ij} represents the degree of how much j^{th} a word in a i^{th} document expresses in sentiment.

Document ID	1	...	m	label
1	x_{11}	...	x_{1m}	positive
...	...	...	...	...
n	x_{n1}	...	x_{nm}	negative

TABLE 2
Data Structure

Three computing processes exist before getting δ_{ij}. The first process trains the context-independent word embeddings x_{ij} into context-dependent word embeddings a_{ij}^P with a self-attention mechanism. The second process fits a logistic regression that takes a_{ij}^P as input and returns u_{ij} as output, the logistic-regression probabilistic score. b is the logistic regression coefficient. The third process converts u_{ij} into δ_{ij} with a sigmoid function as follows:

$$\delta_{ij} = \frac{1}{1 + e^{-u_{ij}}}$$

The reason for using a sigmoid function is that the backpropagation training needs a differentiable function. Yang (2023) also applied two penalty terms, p_{i1} and p_{i2}, to the final δ_{ij}. The penalty term one $p_{i1} = c_1 \sum_{j=1}^{m} (\delta_{ij}(1 - \delta_{ij}))^{1/2}$ pushes all values of δ_{ij} to be close to either 1 or 0 boundaries. c_1 is a hyperparameter that can be tuned. The penalty term two $p_{i2} = c_2 \sum_{j=1}^{m} \delta_{ij}$ controls the density of sentiment words in one document. There are fewer sentiment words if the tuning parameter c_2 is large, and vice versa. The second penalty term is because every document only has a few sentiment words. If all words are used to predict the final document sentiment, lots of noise decreases the prediction performance.

The left red dashed square box in figure 7 represents layer two. The goal of layer two trains the final document sentiment prediction Z_i to match the true sentiment label y_i with the selected words by layer one. The second layer has four processes. The first process trains context-independent word embeddings x_{ij} into context-dependent word embeddings a_{ij}^L. Although this first process in layer two has the same procedure as the first process in layer one, the self-attention training weights are different. This is because the two self-attention transformations have different purposes. The second process fits a logistic regression that takes a_{ij}^L as inputs and returns Z_{ij} as outputs. Z_{ij} is the sentiment score of the j^{th} word in the i^{th} document. β is the logistic regression coefficient. The third process averages all Z_{ij} at the word level in the i^{th} document to get the document-level sentiment score Z_i. The fourth process applies two activation functions to Z_i. The first function is the same sigmoid function used in the first layer. The second function is the indicator, which uses 0.5 as the threshold. If the $S(Z_i)$ is larger than 0.5, the final prediction is 1; otherwise, it is zero. Yang (2023) also applied a third penalty term $p_{i3} = c_3 \sqrt{\sum_{j=1}^{m} (a_{ij^L} \beta)^2}$ to control the absolute magnitude of Z_{ij}. This ensures that Z_{ij} is stable during training.

The last part of this model is the final objective function, as follows

$$Obj(x_{ij}, y_i) = -\frac{1}{n} \sum_{i=1}^{n} [(y_i \log(S(Z_i)) + (1 - y_i) \log(1 - S(Z_i))) + p_{i1} + p_{i2} + p_{i3}]$$

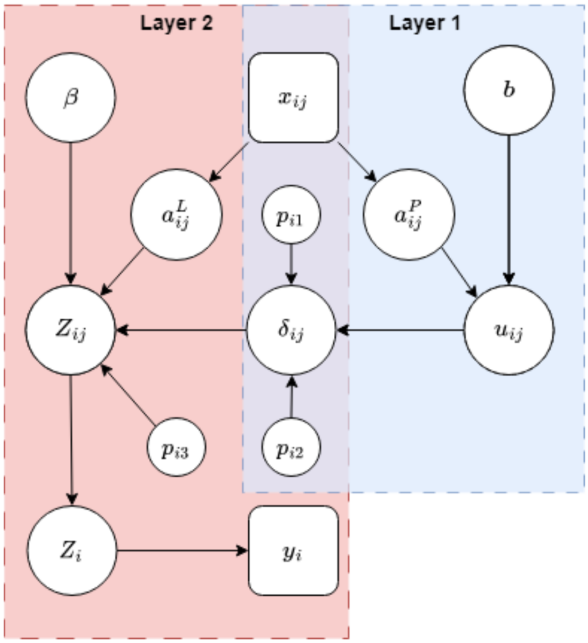


FIG 7. *The Anatomy of AMIC Model*

Because sentiment analysis is a classification task, the cross-entropy loss is applied in the above objective function. This training process is neural-network-based, so stochastic gradient descent with backpropagation minimizes the objective function.

Yang (2023) applied his novel AMIC method to a wine review dataset proposed by Katumullage et al. (2022) and Yang et al. (2022) to evaluate its performance. This wine review dataset has 141,409 reviews. A numeric variable in this dataset called *rating* ranges from 80 to 100. Yang (2023) dichotomized this *rating* variable by labeling numbers less than 90 as negative, otherwise positive. After transforming the response variable, Yang (2023) preprocessed input reviews. He made all reviews have the same length, 150. He also removed words in reviews that didn't exist in **GloVe-300-Wiki** vocabulary. He removed punctuations, and he converted all words to lowercase. He didn't remove stop words and didn't stem all words.

Yang (2023) implemented his AMIC with PyTorch in Python. He split this wine review dataset into training, validation, and testing by a ratio of 18:1:1. The final AMIC model prediction accuracy is reported on the testing set with the parameters that produced the lowest loss on the validation set. The following table 3 compares AMIC with other document-level state-of-the-art neural-network-based models – CNN, BERT, and BILSTM. This table 3 is sorted descending by accuracy. Yang (2023) also compared AMIC with another statistical word-level model, MNIR proposed by Taddy (2013). MNIR was run in R with textir and tm packages. It took a normal CPU computer a few minutes to finish. All other neural-network-based models were run in Google Colab. It took a normal GPU on *Google Colab* between 20 and 30 minutes

to finish the AMIC, CNN and BILSTM models. It took a normal GPU about 2 hours to finish the BERT model. AMIC achieved the highest test accuracy, 89.26%. BERT, BILSTM, and CNN also achieved around test accuracy of 88%, which is very close to the AMIC. MNIR achieved a bit lower test accuracy, 84.68%. However, MNIR took much less time and much fewer resources than neural-network-based models to achieve a word-level interpretable model with only a bit lower accuracy. This arouses us to ask if it is worth applying neural-network-based models to a simple sentiment classification task.

The advantage of AMIC is that it combines statistical interpretability with deep-learning-based high predictive ability. It can identify sentiment words, especially used in wine reviews, that are not used to express sentiment in daily language usage. A self-attention mechanism achieves this context-based word embeddings. The drawback of Yang (2023)'s paper is he only evaluated this novel method on a selected wine review. To show the generality of a novel method, authors need to either simulate many different kinds of data or evaluate a method on a wide range of existing datasets. Another drawback of AMIC is that it doesn't consider the positions of words.

Model	Predictive Accuracy
AMIC	89.26%
BERT	89.12%
BILSTM	88.69%
CNN	88.02%
MNIR	84.68%
MNIR (bigram)	83.45%

TABLE 3
Model Predictive Accuracy

2.3. *The third paper.* In this section, I discussed the third paper, "Learning the difference that makes a difference", written by Kaushik, Hovy and Lipton (2019). The main point of this paper is that some researchers found that neural network-based models only use surface-level patterns or associations of data to finish the task. Kaushik, Hovy and Lipton (2019) found that this phenomenon greatly affected the generalizability of "sentiment analysis" models and "natural language inference" models. Models that are trained on the original dataset perform poorly on the counterfactual-revised counterparts. One reason for this phenomenon, I assumed, is that neural-network-based models are "black boxes," which makes researchers hardly control what truly works in such complicated models under the hood. Another reason is that neural-network-based models are so easy to overfit that they perform poorly on the out-of-sample data. So, he proposed a method that makes natural language processing models more versatile. The novel method allows humans to create new observations with counterfactual texts and labels according to the original observations. This creation requires that new observations make sense and edits are minimized. Kaushik, Hovy and Lipton (2019) elucidated this human-in-the-loop data augmentation by figure 8. Then, he retrained

models on combining the original and newly created datasets. As a result, he found that such models are less sensitive to surface-level patterns. Such models achieved a bit lower accuracy on the original dataset than those trained only on the original one. However, it performs much better on the counterfactual-revised counterparts than those trained only on the original. He revised an IMDB movie review dataset proposed by Maas et al. (2011) for sentiment classification. This IMDB review dataset was split into 1707 training, 245 validation, and 488 tests. He revised the "Stanford Natural Language Inference Corpus" proposed by Bowman et al. (2015), the popular benchmark data for natural language inference tasks (NLI). He randomly sampled 1750 training, 250 validation, and 500 tests from the SNLI. Then, he asked workers to revise the premises and hypothesis, respectively. Lastly, he got 6664 pairs of training, 800 pairs of validation, and 1600 pairs of tests. Sentiment Analysis is usually a binary classification task, positive or negative. NLI is usually a 3-way classification task. NLI classifies a pair of premises and hypotheses into three relations – entailment, contradiction, and neutral.

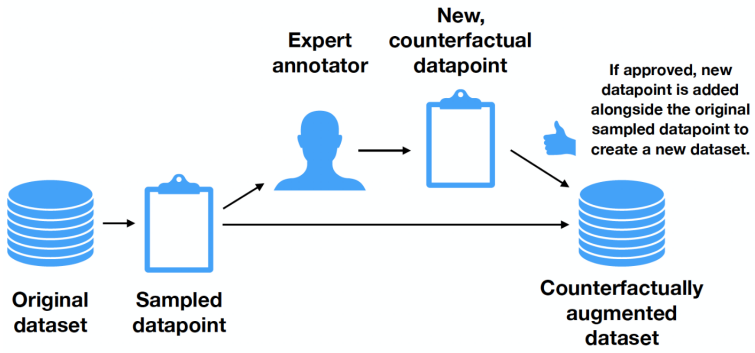
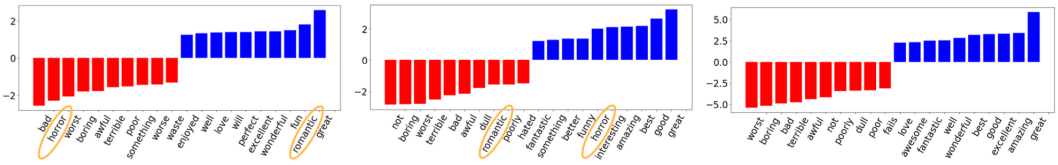


FIG 8. *The process of augmenting dataset*

Kaushik, Hovy and Lipton (2019) evaluated his novel method with five models – Support Vector Machine (SVM), Naïve Bayes Model (NB), BILSTM, ELMo-LSTM, and BERT. They used *scikit-learn* package to run the SVM and NB model for sentiment analysis. They used *PyTorch* package to run BILSTM, ELMo-LSTM, and BERT model. After evaluations, they found it worth paying extra to let models learn the surface-level and deep-level patterns by augmenting data with counterfactual instances. They also evaluated their models on out-of-domain datasets, including Amazon reviews, the Twitter sentiment dataset, and Yelp reviews. They showed that models trained on the counterfactual-augmented dataset performed better than those trained only on the original dataset. For example, the BILSTM model trained on the combined dataset (the original and the revised) achieved 82.7% accuracy on the Amazon review dataset for sentiment analysis. However, the BILSTM model trained only on the original dataset achieved 65.9% accuracy on the same Amazon review dataset. The full evaluation result tables are displayed in Kaushik, Hovy and Lipton (2019)'s paper. The advantage of their novel method is that counterfactual-augmented datasets push models

to learn the non-surface-level texture and patterns. For instance, in the IMDB movie review dataset, *"horror"* and *"romantic"* are two movie genre words. However, the SVM models trained only on the original or revised data selected these two movie genre words as sentiment words. This shows models trained on a partial dataset learn something wrong in sentiment analysis. But, when the SVM model was trained on the combined dataset, *"horror"* and *"romantic"* were not selected to be sentiment words. Instead, words such as *"poorly"* and *"awesome"* were selected. Kaushik, Hovy and Lipton (2019) elucidated the above phenomenon "Learn the difference what makes a difference" in the figure 9.



(a) Trained on the original dataset (b) Trained on the revised dataset (c) Trained on combined dataset

FIG 9. *Different sentiment words selected by SVM on different kinds of data*

The same "drop" phenomenon happened to the NLI task. For example, the BERT model trained only on the 8,300 original data achieved 77.8% accuracy on the original data. However, it only achieved 44.6% accuracy on the revised-premise dataset and 66.1% accuracy on the revised-hypothesis dataset. However, the BERT model trained on the combined dataset achieved 73.5% accuracy on the original, 64.6% accuracy on the revised-premise data, and 69.6% accuracy on the revised-hypothesis data.

Overall, counterfactual-augmented data helps models learn the non-surface-level pattern. This makes them versatile and general in sentiment analysis and natural language inference tasks.

3. Synthesis of Core Papers. After discussing the above three core papers, I summarized and synthesized them in this section. The relation of these papers is a linear sequence. The first paper, "**Text as Data**" written by Gentzkow, Kelly and Taddy (2019), aims to survey methods of representing text into numeric data. This paper gives us a collection of ways to prepare data. The second paper, "**Interpretable Sentiment Analysis Using the Attention-Based Multiple Instance Classification Model**" written by Yang (2023), proposed an attention-based multiple instance classification model (AMIC). He used two methods of representing text as data, discussed in the first paper Gentzkow, Kelly and Taddy (2019). The first method counts the n-gram tokens in documents, forming the document-term matrix. He used this document-term matrix to run the baseline sentiment analysis model – MNIR- discussed in the first paper. The second method is **GloVe-300-Wiki**, which represents words into context-independent embeddings. He applied this method in his novel proposed AMIC. Gentzkow, Kelly and Taddy (2019) also discussed many applications in sentiment analysis. Gentzkow, Kelly and Taddy (2019) thought in social science and economics, it is useful

and important to have model interpretability and causal inference. However, they found most state-of-the-art neural-network-based models don't provide any interpretations, though they can achieve much higher predictive accuracy than traditional statistical methods. Yang (2023) positively addressed their issue by proposing AMIC, which can give word-level sentiment scores. This neural network method provides accessibility and interoperability. The third paper, "**Learning the Difference that makes a difference with Counterfactually-Augmented Data**" written by Kaushik, Hovy and Lipton (2019), pointed out that some NLP models only learn the surface-level patterns of data to make predictions in sentiment analysis and natural language inference. This drawback makes these NLP models perform poorly on the out-of-sample and sensitive to these learned surface-level patterns. However, Yang (2023) didn't augment the *wine review* data with counterfactual instances for evaluating his novel AMIC on sentiment analysis. This makes us think about if Yang (2023)'s AMIC for sentiment analysis suffers the same issue as Kaushik, Hovy and Lipton (2019) pointed out. As a result, the third paper can be used to test if AMIC has the same issue. If so, the third paper is a future direction for augmenting the *wine review* dataset with counterfactual instances. Then, AMIC will be trained on this augmented data to generalize and make it less sensitive to the surface-level patterns. However, augmenting *wine review* data is a big challenge. It requires many human efforts to revise the original data manually to some scale level. This costs a lot of money and time to finish. Whether it is worth augmenting data depends on the application and its purpose. If a wine company wants to monitor their wine product quality by their customers' reviews and further improve their wine, I think for this company, it's worth investing money and resources to augment data. However, for researchers, because of limited resources, creating a few toy counterfactual examples and testing the performance of AMIC on it is enough.

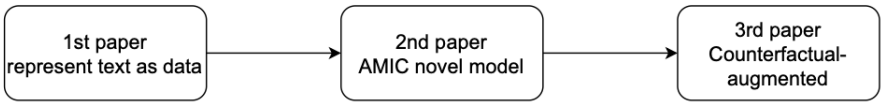


FIG 10. *The linear sequence of three core papers*

From the results of the second and the third papers, before experiments, I assume counterfactual instances of *wine review* could further improve the generalizability of the AMIC model. However, I think the degree of improvement is limited. I explained my assumption as follows. In Figure 10, I discussed that models trained only on the original dataset *IMDB* selected movie genre words such as *"horror"* and *"romantic"* as highly weighted sentiment indicators. But models trained on the counterfactual-augmented dataset replaced movie genre words with exact sentiment-expressing words such as *"poorly"* and *"awesome"*. In the Yang (2023)'s paper, he displayed the AMIC top 50 positive sentiment words in table 11. I found that words that express the actions of tasting wines, such as *"drips"* and *"soak"*, were picked up by AMIC model as top positive

sentiment words. These action words are usually considered neutral in the sentiment analysis. However, most of the top 50 sentiment words in this table 11 expressed positive. Retraining the AMIC model on a counterfactual-augmented *wine review* could help it learn more non-surface-level patterns and generalize well to out-of-domain data, but the degree of improvement is determined by how many counterfactual instances are created and the quality of them.

gorgeous	beautiful	ethereal	beautifully	gloriously
gorgeously	thoroughly	drips	beauty	impeccable
amazingly	exquisite	strikingly	sumptuous	cognac
burgundy	breed	velvet	cascading	haunting
seductive	finely	stuffed	soak	lovely
soaked	perfectly	deliciously	brilliantly	impeccably
wonderful	drip	luxuriant	glistening	silk
truffle	charms	brunello	soothing	carpet
champagne	perfume	seductively	fabric	unobtrusive
sings	swirl	stained	wonderfully	elegance

Note: size of a word is proportional to it sentiment score

FIG 11. *AMIC's top 50 positive sentiment words*

4. Technical Details. After discussing each core paper and synthesizing them, researchers must be able to reproduce and validate the results of novel methods and validate them. Belz et al. (2021) found that in the natural language understanding area, only a small percentage of papers' results can be reproduced to have the same results under the same conditions shown in the original papers. They also found that the reproduction results were usually worse than reported in the original papers if a discrepancy happened. In natural language understanding, now there is no clear definition of reproducibility and how researchers should measure it. Although there is no clear definition, my measurement is to try to get the test accuracy as close to the report as possible.

I reproduced the AIMC model performance results for sentiment analysis in this section. The table I reproduced in the second paper, written by Yang (2023), is shown on Page 9. His table is shown in the below figure 12. I reordered each model's predictive accuracy in descending order in table 3 when I discussed his paper in section two.

I explained the whole process of my reproduction of his novel AMIC method and some other models shown in the table 12. Regarding AMIC, luckily, when I contacted the author Chenyu Yang about reproducing his results, he shared me with his Jupyter Notebooks that included his codes of building his AMIC and some simple inference of results. He also gave me the dataset used to evaluate AMIC in the second paper. The dataset is *wine reviews*, which has 141,409 observations and was collected from Wine Spectator from

	Model	Predictive Accuracy
Word-level SA models	AMIC	**0.8926**
	MNIR	0.8468
	MNIR (with bigram)	0.8345
Neural Network SA models	CNN	0.8802
	BILSTM	0.8869
	BERT	0.8912

FIG 12. *Compare AMIC performance with other models*

2005 to 2016. This dataset was shared in an Excel format. He also shared the **GloVe** word embeddings of 300 dimensions. All the above files were shared via SMU Box.

Regarding AMIC, I used the Google Colab to run his codes. T4 GPU and High-RAM were used to compute. His AMIC was implemented in PyTorch. The torch version is *"2.1.0+cu121"*. The *wine reviews* data was stored in Google Drive, and it was read in colab into a pandas format. The dataset has ten variables, but only *rating* and *Clean_desc* variables were used. He then shuffled the data with a random seed = 42. Then, there are many preprocessing steps to represent review descriptions in data. First, reviews were tokenized by Tokenize, and a dictionary of mapping token IDs to words was created. Second, he initialized a word-embedding dictionary with the **Glove** file. There are 400,000 word embeddings in this dictionary. He then selected word embeddings that uniquely existed in the reviews. Then, he got an embedding matrix of 10,854 by 300. He also created a dictionary that maps token IDs to words. Because the *rating* is a numeric continuous variable, he dichotomized this variable by assigning $rating > 89$ as one and $rating <= 89$ as zero to conduct a binary sentiment analysis. Whenever we build models to predict a classification task, we must inspect the balance of classes in the response variable. I plotted a bar plot to observe the distribution of this dichotomized rating variable in figure 13. From this figure, I found that the ratio of zero to one is close to 2:1. Usually, the ideal ratio for binary classification is 1:1. However, this 1:1 ratio is practically difficult to get. Empirically, a ratio of 3:2 can still produce a good model under careful building. The 2:1 of *wine reviews* is still acceptable, but we must be mindful of a bit of class imbalance. Yang (2023) defined a function to fix all reviews of the same length of 100. This is realized by padding short reviews with prefixed zeros or truncating long reviews at the tail. It's common to align all documents with the same length in natural language understanding. The first reason is that stochastic gradient descent in backpropagation usually uses one batch of datasets, which involves a lot of matrix multiplication and matrix pairwise operations. Different sequence lengths in one batch probably complicate the implementations and further slow the training. The second reason is that some natural language models even require input sequence lengths to be the same. In his implementation, he didn't remove stop words because he thought the first layer of his attention-based models was trained to select sentiment words. He didn't stem words because he concluded that different base word forms in wine reviews sometimes carry different sentiments. For instance, *"acidity"*

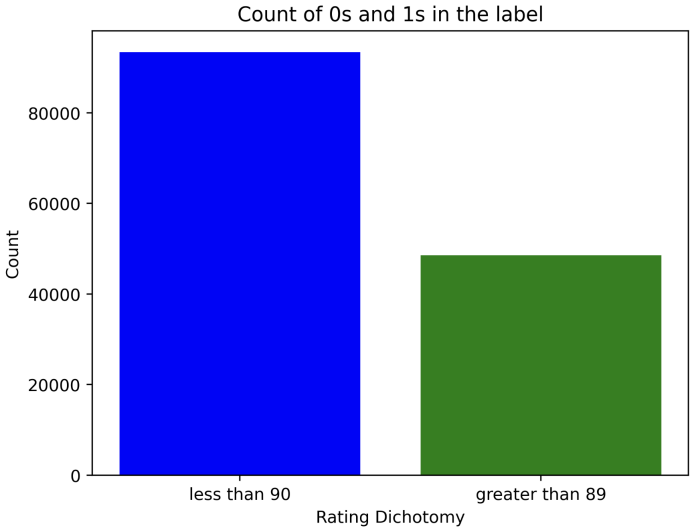


FIG 13. *The bar plot of the distribution of this dichotomized rating*

means positive in a review, *"The acidity of this wine is balanced"*. However, *"acidic"* means negative in a review, *"This wine is too acidic"*. So, stemming all words decreases the test accuracy.

After preprocessing reviews, Yang (2023) split the dataset into training, validation, and testing sets. The split ratio is 18:1:1. Then, three sets were converted into tensor format with torch.from_numpy. He used DataLoader to create a batch of 64 and shuffle each batch.

After dealing with data, he defined a few Python classes to implement his AMIC. The first class is **SelfAttention** inherited from a base class named **nn.Module**. This base class serves all neural network modules, and all self-defined neural network methods should subclass it. This **SelfAttention** class uses an 8-head self-attention method that includes *query*, *key*, and *value*. In natural language understanding, a *query* usually refers to a task or question. A *key* refers to the relevant important information to this *query*. A *value* refers to a corpus of documents, books, articles, and so on. The second class is **tiedLinear** inherited from **nn.Module**. This class is defined to reduce model complexity and prevent overfitting using the same weight matrix for the input and output layers. The third class is **PositionalEncoding**. This class adds the word position information into the word embeddings. This is critical and helpful for sequence data, especially for natural language understanding. The fifth class is **Mask_block**. This class incorporates class **SelfAttention**, **tiedLinear**, dropout for regularization, and p_1 & p_2 for penalty terms. The sixth class is **Senitment_block**. This uses **SelfAttention**, **tiedLinear**, **PositionalEncoding**, and dropout regularization to compute word-level context-based attentions. The last layer calculates the average word sentiment scores in one review to predict the document-level sentiment. The seventh class is **Synthesizer**. This class synthesizes the **SelfAttention** and **tiedLinear**. The eighth class is **Embeds**. This class incorporates the **PositionalEncoding** information into the pre-trained input word embeddings.

After defining all the above essential classes, he defined the number of layers to be 2, the number of heads to be 10, the dimension of hidden layers to be 300, and fixed the random seeds for reproducibility. Then, he initialized objects of **Embeds, Snetiment_block, Mask_block, and Synthesizer**. Now, he needed to set up the hyperparameters in the training process. There are many hyperparameters in this model. In his training process, he used the most popular gradient-based optimization algorithm for his stochastic objective function, Adam. Kingma and Ba (2014) proposed Adam, and their paper has been cited 164,147. Adam is very well-known for its fast convergence and adaptive learning rating. Yang (2023) set up 0.0002 learning rates for four objects. He also used scheduler function to decay learning rates of parameters by gamma every a few steps. Unfortunately, the default numbers for these hyperparameters in his code didn't reproduce his 89.26% test accuracy reported in his paper. Then, I spent some time tuning these hyperparameters to reach his results. I tuned the step size to be 2 and the gamma to be 0.65 in the above scheduler. Yang (2023) chose Binary Cross Entropy to calculate the loss between predictions and true labels. Yang (2023) used 10 epochs in his code. However, I found that 10 epochs were insufficient because, at the 10th epoch, the rate of training accuracy wasn't flat. I set up 15 epochs for this training process. After discussing with Yang (2023) that I couldn't reproduce his reported accuracy, he told me that c_1 in the first penalty term was an important hyperparameter that greatly affected the performance. The default training codes that he sent to me had a few bugs. The bug occurred in the **Sentiment_block** object. The definition of this object has three outputs in the forward function. However, in his code, he only specified one output to receive the outcome from this object. Because his training code is very long in one code chunk, Python only threw an error that the dimensions didn't match between one matrix and another, I spent a long time locating this bug by printing the intermediate matrix dimensions. The model was saved whenever the validation loss of this model was lower than a predefined constant. Yang (2023) didn't describe what this predefined constant is. After reviewing his codes, I assume this predefined constant is the best training model loss he has reached. In neural network training, plotting the changes in accuracy and loss against epochs is common and important. However, Yang (2023) didn't do this in his shared code. I added a new line of codes to record each epoch's validation accuracy, validation loss, and training loss for plotting purposes. The whole training process took me 13 minutes to finish. I plotted the average validation accuracy for each epoch in figure 14. I observed that the average validation accuracy increased sharply between Epoch 1 and Epoch 5. However, between Epoch 6 and Epoch 12, the average validation accuracy slowly increased. After Epoch 13, this accuracy stayed the same almost. I also plotted each epoch's average training and validation loss in figure 15. I observed that the validation loss decreased considerably before Epoch 4. After Epoch 5, it decreased very slowly and stayed the same. However, the training loss fluctuated very often. The final best saved model achieved 88.99% training accuracy. It achieved 88.66% validation accuracy. However, it only achieved 88.89% test accuracy. My reproduction test accuracy, 88.89%, is slightly lower than Yang (2023)'s reported one, 89.26%. In addition to this implementation of AMIC, he conducted some inference analysis

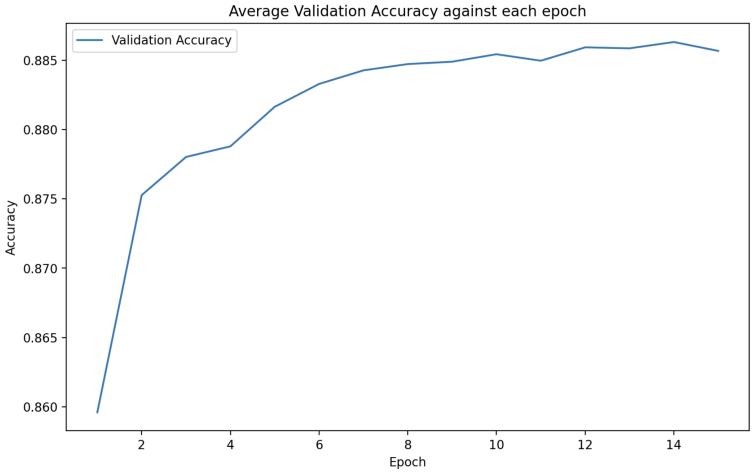


FIG 14. *The average validation accuracy of each epoch*

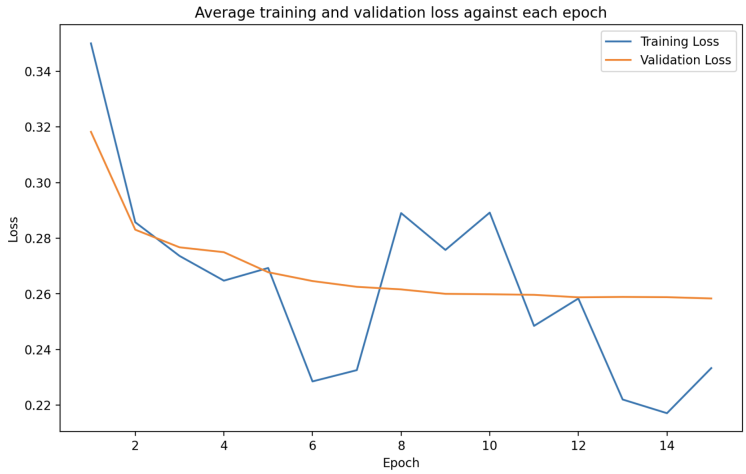


FIG 15. *The average training and val loss of each epoch*

based on his trained models in his shared later codes. However, one code chunk must import external data files; most following codes depend on these files. He didn't share this with me. My edited version of Yang (2023)'s shared codes is available for further inspection at the colab Jupyter Notebook.

Next, I reproduced his BERT model results because compared to CNN and BILSTM, BERT is the most state-of-the-art natural language understanding model. Unfortunately, Yang (2023) didn't share his BERT model codes. I had to write a BERT model from scratch and fine-tune it on the same *wine review* dataset. In addition to all Python packages used in the AMIC model, the core package used to build a BERT model is the transformers package developed by the Hugging Face team. I didn't apply the same preprocessing procedures that Yang (2023) applied to his AMIC model. Instead, I applied the most basic pre-trained BERT

tokenizer, bert-base-uncased, to the *wine reviews*. *bert-base-uncased* proposed by Devlin et al. (2018) was pre-trained on a large corpus of English texts in a self-supervised way and doesn't differentiate cases. This tokenizer produces two parts for the training process. One part is *input_ids*, and another is *attention_masks*. I split the data into training, validation, and testing with the same ratio and random seed used in AMIC to ensure the splitting is the same as AMIC. Then I applied *TensorDataset* and *DataLoader* to prepare them for training. I shuffle each batch using *RandomSampler* and *SequentialSampler*. The batch size is 64, the same as AMIC. I used AdamW with a learning rate, 2×10^{-5}, as the optimization algorithm. I used BCEWithLogitsLoss as the loss function. This loss function is more numerically stable by combining a sigmoid layer with a BCELoss function into one class. Because this is a binary classification task, I used BertForSequenceClassification class to initialize the model with the *bert-base-uncased* and set *num_labels* equal to one. I used "torch.nn.utils.clip_grad_norm_" to cut off the gradient of the model parameters during training. This can prevent them from exploding. This ensures training is stable. I only used 2 epochs for training because the validation loss started decreasing from the second epoch. I used the same platform, colab, resources, T4 GPU, and high-Ram. It took 1 hour and 4 minutes to finish. I plotted the validation accuracy, loss, and training loss in figure 16 and figure 17. The validation accuracy decreased from Epoch One to Epoch Two in the figure 16. The validation loss increased from Epoch One to Epoch Two in the figure 17. I applied the trained model to the test set and found that the test accuracy was 88.75%. This 88.75% test accuracy is also slightly lower than the test accuracy Yang (2023) reported, 89.12%. However, my reproduced AMIC test accuracy, 88.89%, is higher than my defined BERT test accuracy, 88.75%, though they are essentially very close. For a more robust comparison result, k-fold cross-validation is desirable because it gives us the mean and standard deviation of test accuracy. However, I didn't do it because Yang (2023) didn't apply it in his original code. He only reported the best model he had reached, and so did I. From this comparison, I found the biggest advantage of AMIC was it provided word-level interpretability and achieved comparable predictive performance with much less running time and resources. My code for the BERT model is available for further reproduction and inspection at the colab Jupyter Notebook.

5. Future Directions. This section explored some future directions beyond Yang (2023)'s work. There are many ways to extend his work. The most straightforward direction is that I inspect if his method suffers from learning some surface-level patterns to make sentiment analysis. My idea was inspired by the Kaushik, Hovy and Lipton (2019)' work. As I discussed in section two, they found that some natural language understanding models usually classify sentiments by spurious textures, and they proposed a method to augment the original data with counterfactual instances to retrain models. Kaushik, Hovy and Lipton (2019) published their IMDB movie review dataset for the sentiment analysis at their GitHub. I pulled their original, revised, and combined data from their GitHub to experiment with them on the Yang (2023)'s AMIC.

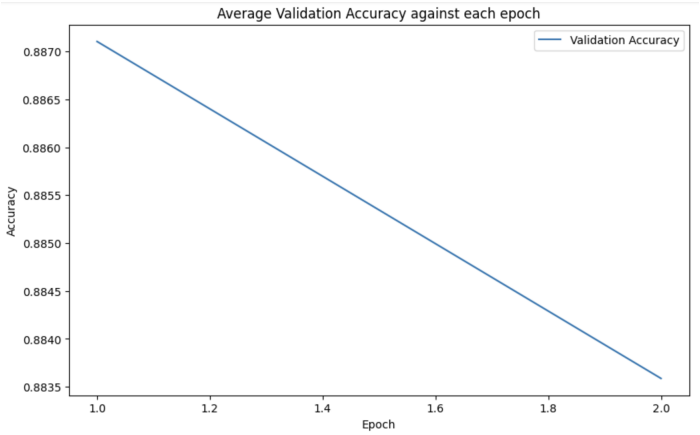


FIG 16. *The average validation accuracy of each epoch*

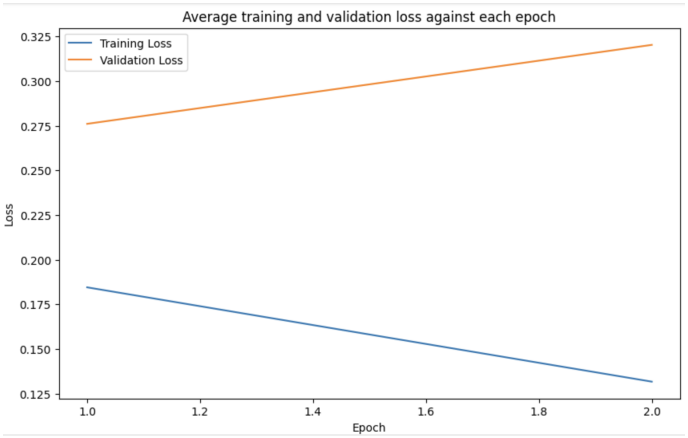


FIG 17. *The average training and val loss of each epoch*

I used the same platform, Google Colab, and their T4 GPU for this experiment. Kaushik, Hovy and Lipton (2019) have split their IMDB dataset into training, validation, and testing sets. The original training set has 1707 observations, the original validation set has 245 observations, and the original test set has 488 observations. I printed the head of this IMDB dataset in my Jupyter notebook and placed the screenshot of it in the figure 18. Then, I applied all the same preprocessing procedures, including tokenizing reviews, creating word embeddings for unique words, removing words, and calculating review lengths. I plotted a bar plot to inspect the ratio between negative and positive sentiments in the original IMDB dataset in Figure 19. The ratio is 1:1, an ideal situation for a binary classification. I increased the padded length for all movie reviews to 350 instead of 100 in Yang (2023)'s *wine review* because this IMDB dataset has much fewer observations than *wine review*. I used the same *TensorDataset* and *DataLoader* to prepare the IMDB data for training in PyTorch. In the *DataLoader*, I increased the batch size to 128 instead of 64 after

	Sentiment	Text
0	Negative	Long, boring, blasphemous. Never have I been s...
1	Negative	Not good! Rent or buy the original! Watch this...
2	Negative	This movie is so bad, it can only be compared ...
3	Negative	Spanish horrors are not bad at all, some are s...
4	Negative	I've seen about 820 movies released between 19...

FIG 18. *The head of IMDB dataset*

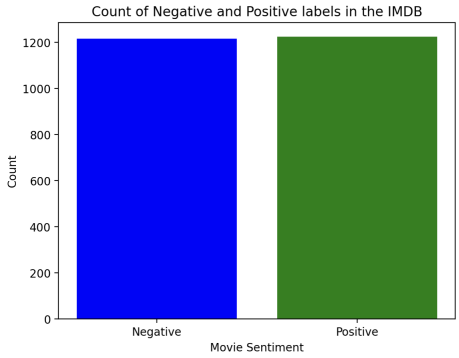


FIG 19. *The bar plot of negative and positive sentiments in the original IMDB*

tuning the batch size parameter. The size of the final embedding matrix for all words is 21,196 by 300. I applied all the same Yang (2023)'s defined classes in his AMIC to the IMDB data. However, I encountered a big challenge during the training process. Since the data was changed completely, the original default hyperparameters achieved around 50% validation accuracy, a random guess. Then, I spent at least 8 hours tuning the hyperparameters of AMIC, including four learning rates, step sizes, gamma (decay ratio), c_1, c_2 penalty constants, and the number of epochs. I set up **Embeds**'s learning rate to be 0.05, **Sentiment_block**'s learning rate to be 0.09, **Mask_block**'s learning rate to be 0.07, and **Synthesizer**'s learning rate to be 0.06. I set up all the learning rates to decay by 0.7 for every 1-step size of each epoch. I set up 30 epochs in total. I set up the c_1 penalty constant to be 0.001. I changed the way I saved the best AMIC model during training. Yang (2023) saved the best model whenever the validation loss was lower than a predetermined constant. I found this way was a pitfall because a high validation accuracy could have either low or high validation loss. In light of this, I saved the best model whenever the validation accuracy was higher than 84% because the validation accuracy is not stable as shown in the figure 20, making me not to adopt an early stopping strategy. After about 2 minutes of training, I plotted the validation accuracy changes against each epoch and loss for the training and validation sets in the figure 20 and 21. In the figure 20, the validation

accuracy kept increasing though fluctuating. In figure 21, the training loss decreased sharply before Epoch 5 but decreased very slowly after Epoch 5. The validation loss decreased before Epoch 5 but increased after Epoch 5, which suggested this training may be overfitting. The final trained model achieved 97.9% training accuracy, 78.91% validation accuracy, and 77.6% test accuracy.

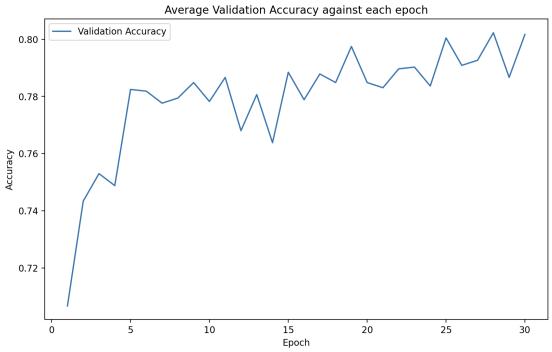


FIG 20. *Validation Accuracy on the original IMDB data*

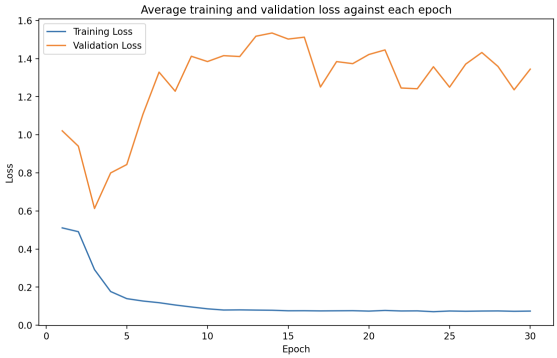


FIG 21. *Validation Loss on the original IMDB data*

Then I imported the revised IMDB dataset. The training set had 1,707 observations, the validation set had 245 observations, and the test set had 488 observations. I applied all the same preprocessing procedures to this revised dataset. However, the trained model on the original IMDB dataset only achieved 52.7% accuracy on the revised training set. Kaushik, Hovy and Lipton (2019) predicted this phenomenon and proved it in the figure 22. For example, the Bi-LSTM model trained only on the original IMDB achieved 79.3% accuracy but only 55.7% accuracy on the revised IMDB. I can conclude that Yang (2023)'s AMIC still suffers from only learning the spurious textures in the sentiment analysis.

Training data	SVM		NB		ELMo		Bi-LSTM		BERT	
	O	R	O	R	O	R	O	R	O	R
Orig. (1.7k)	**80.0**	51.0	**74.9**	47.3	**81.9**	66.7	**79.3**	55.7	**87.4**	82.2
Rev. (1.7k)	58.3	**91.2**	50.9	**88.7**	63.8	**82.0**	62.5	**89.1**	80.4	**90.8**
Orig. – Edited	57.8	–	59.1	–	50.3	–	60.2	–	49.2	–
Orig. & Rev. (3.4k)	83.7	**87.3**	**86.1**	**91.2**	**85.0**	**92.0**	**81.5**	**92.0**	88.5	**95.1**
Orig. (3.4k)	**85.1**	54.3	82.4	48.2	82.4	61.1	80.4	59.6	**90.2**	86.1
Orig. (19k)	**87.8**	60.9	84.3	42.8	86.5	64.3	86.3	68.0	93.2	88.3
Orig. (19k) & Rev.	**87.8**	**76.2**	**85.2**	**48.4**	**88.3**	**84.6**	**88.7**	**79.5**	**93.2**	**93.9**

FIG 22. *The accuracy of IMDB on the original and revised data with different training data*

Next, I investigated if models trained on the combined original and revised data improved the accuracy of the original and revised data separately. I trained the AMIC on the combined data by using all the same preprocessing procedures to the movie reviews. After tuning hyperparameters of AMIC models, I finally

set up **Sentiment_block**'s learning rate to be 0.07 and other classes' learning rates to be 0.05. The step size was one, and the gamma was 0.9. I used 30 epochs to train AMIC. I plotted the validation accuracy changes and loss in the figure 23 and 24. This AMIC model learned a lot before Epoch 14, and the validation loss showed it started overfitting after Epoch 14.

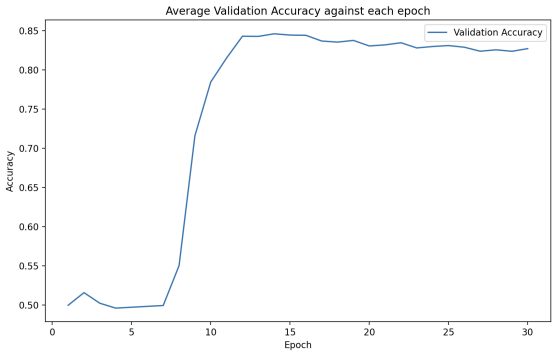


FIG 23. *Validation Accuracy on the combined data*

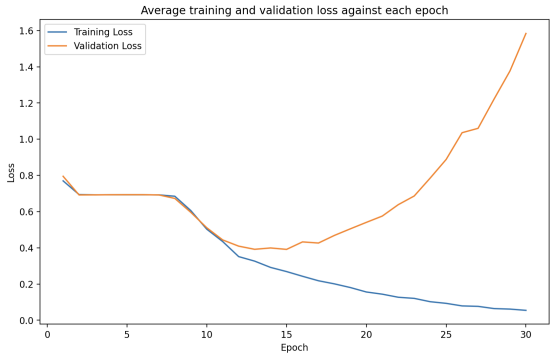


FIG 24. *Validation Loss on the combined data*

I made a table 4 to report the test accuracies trained on the original IMDB data, and the combined IMDB data with the AMIC model. In the table 4, "O" represents the original and "R" represents the revised. The AMIC model trained on the original achieved 77.6% test accuracy on the original IMDB data while only achieving 52.7% on the revised data. The AMIC model trained on the combined IMDB of the original and the revised data achieved 80.99% test accuracy on the original and 86.46% on the revised. This suggests that Yang (2023)'s novel AMIC method also suffers from learning some spurious patterns. However, we can augment the sentiment data with counterfactual instances to train AMIC to be less dependent on the spurious patterns in the sentiment analysis. This induces a future direction for creating a revised IMDB dataset. I plan to replace the sentiment words picked up by AMIC models in each review with similar-meaning words of the same degree and sentiment. For instance, "beautiful" can be replaced with "gorgeous". Then I plan to retrain the combined the dataset to improve the AMIC's generality. The good side of AMIC is that it can pick up the sentiment words and their scores, automating the revising process instead of manually time-consuming revising by humans.

Training data	AMIC		SVM		NB		ELMo		Bi-LSTM		BERT	
	O	R	O	R	O	R	O	R	O	R	O	R
Orig. (1.7k)	77.6	52.7	80.0	51.0	74.9	47.3	81.9	66.7	79.3	55.7	87.4	82.2
Orig. & Rev. $(3.4k)$	80.99	86.46	83.7	87.3	86.1	91.2	85.0	92.0	81.5	92.0	88.5	95.1

TABLE 4
Test Accuracy on the different dataset with AMIC model

The final future direction is to build a revised AMIC to make continuous numeric predictions instead of sentiment analysis. Yang (2023) dichotomized the *rating* variable, which lost a lot of information. Instead, a regression version of AMIC that produces numeric rating predictions is more suitable for the wine industry.

REFERENCES

BELZ, A., AGARWAL, S., SHIMORINA, A. and REITER, E. (2021). A Systematic Review of Reproducibility Research in Natural Language Processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume* (P. MERLO, J. TIEDEMANN and R. TSARFATY, eds.) 381–393. Association for Computational Linguistics, Online. https://doi.org/10.18653/v1/2021.eacl-main.29

BENGIO, Y., DUCHARME, R. and VINCENT, P. (2000). A neural probabilistic language model. *Advances in neural information processing systems* **13**.

BOSER, B. E., GUYON, I. M. and VAPNIK, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* 144–152.

BOWMAN, S. R., ANGELI, G., POTTS, C. and MANNING, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (L. MÀRQUEZ, C. CALLISON-BURCH and J. SU, eds.) 632–642. Association for Computational Linguistics, Lisbon, Portugal. https://doi.org/10.18653/v1/D15-1075

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

CANDÈS, E. J., WAKIN, M. B. and BOYD, S. P. (2008). Enhancing Sparsity by Reweighted l1 Minimization. *Journal of Fourier Analysis and Applications* **14** 877–905. https://doi.org/10.1007/s00041-008-9045-x

CHEN, T. and GUESTRIN, C. (2016). XGBoost: A Scalable Tree Boosting System. *CoRR* **abs/1603.02754**.

CYBENKO, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* **2** 303–314. https://doi.org/10.1007/BF02551274

DEVLIN, J., CHANG, M., LEE, K. and TOUTANOVA, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* **abs/1810.04805**.

DOYLE, D. Ranks NL. https://www.ranks.nl/stopwords.

GENTZKOW, M., KELLY, B. and TADDY, M. (2019). Text as Data. *Journal of Economic Literature* **57** 535-74. https://doi.org/10.1257/jel.20181020

HIEMSTRA, D. (2009). *N-Gram Models* In *Encyclopedia of Database Systems* 1910–1910. Springer US, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_935

HOCHREITER, S. and SCHMIDHUBER, J. (1997). Long Short-Term Memory. *Neural Comput.* **9** 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

HOERL, A. E. and KENNARD, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* **12** 55–67.

HOTELLING, H. (1933). Analysis of a complex of statistical variables with principal components. *J. Educ. Psy.* **24** 498-520.

KATUMULLAGE, D., YANG, C., BARTH, J. and CAO, J. (2022). Using Neural Network Models for Wine Review Classification. *Journal of Wine Economics* **17** 27–41. https://doi.org/10.1017/jwe.2022.2

KAUSHIK, D., HOVY, E. H. and LIPTON, Z. C. (2019). Learning the Difference that Makes a Difference with Counterfactually-Augmented Data. *CoRR* **abs/1909.12434**.

KIM, Y. (2014). Convolutional Neural Networks for Sentence Classification. *CoRR* **abs/1408.5882**.

KINGMA, D. P. and BA, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

KRIZHEVSKY, A., SUTSKEVER, I. and HINTON, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems* (F. PEREIRA, C. J. BURGES, L. BOTTOU and K. Q. WEINBERGER, eds.) **25**. Curran Associates, Inc.

LIU, Y., SUN, C., LIN, L. and WANG, X. (2016). Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention. *CoRR* **abs/1605.09090**.

MAAS, A. L., DALY, R. E., PHAM, P. T., HUANG, D., NG, A. Y. and POTTS, C. (2011). Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (D. LIN, Y. MATSUMOTO and R. MIHALCEA, eds.) 142–150. Association for Computational Linguistics, Portland, Oregon, USA.

MACCARTNEY, B. (2014). Understanding Natural Language Understanding Technical Report, Stanford University, Stanford, CA.

MANNING, C. D. and SCHÜTZE, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

MCCULLOCH, W. S. and PITTS, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* **5** 115–133. https://doi.org/10.1007/BF02478259

MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. and DEAN, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *CoRR* **abs/1310.4546**.

MURDOCH, W. J. (2019). Interpretable deep learning for natural language processing, PhD thesis, UC Berkeley.

NELDER, J. A. and WEDDERBURN, R. W. M. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society. Series A (General)* **135** 370–384.

PENNINGTON, J., SOCHER, R. and MANNING, C. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (A. MOSCHITTI, B. PANG and W. DAELEMANS, eds.) 1532–1543. Association for Computational Linguistics, Doha, Qatar. https://doi.org/10.3115/v1/D14-1162

REID, S. W., MCKENNY, A. F. and SHORT, J. C. (2023). *Synthesizing Best Practices for Conducting Dictionary-Based Computerized Text Analysis Research*. In *Methods to Improve Our Field. Research Methodology in Strategy and Management* **14** 43–78. Emerald Publishing Limited. https://doi.org/10.1108/S1479-838720220000014004

SPARCK JONES, K. (1972). A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL. *Journal of Documentation* **28** 11–21. https://doi.org/10.1108/eb026526

TADDY, M. (2013). Multinomial Inverse Regression for Text Analysis. *Journal of the American Statistical Association* **108** 755-770. https://doi.org/10.1080/01621459.2012.734168

TIBSHIRANI, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* **58** 267–288.

TURNEY, P. D. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *CoRR* **cs.LG/0212032**.

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L. and POLOSUKHIN, I. (2017). Attention Is All You Need. *CoRR* **abs/1706.03762**.

WOLD, H. (2004). *Partial Least Squares*. https://doi.org/10.1002/0471667196.ess1914

WOLF, T., DEBUT, L., SANH, V., CHAUMOND, J., DELANGUE, C., MOI, A., CISTAC, P., RAULT, T., LOUF, R., FUNTOWICZ, M. and BREW, J. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR* **abs/1910.03771**.

XIONG, D. (2021). Bayesian multiple instance learning with application to cancer detection using TCR repertoire sequencing data, PhD thesis, Southern Methodist University.

YANG, C. (2023). INTERPRETABLE SENTIMENT ANALYSIS USING THE ATTENTION-BASED MULTIPLE INSTANCE CLASSIFICATION MODEL, PhD thesis, Southern Methodist University, Dallas, TX.

YANG, C., BARTH, J., KATUMULLAGE, D. and CAO, J. (2022). Wine Review Descriptors as Quality Predictors: Evidence from Language Processing Techniques. *Journal of Wine Economics* **17** 64–80. https://doi.org/10.1017/jwe.2022.3

ZHANG, L., WANG, S. and LIU, B. (2018). Deep Learning for Sentiment Analysis : A Survey. *CoRR* **abs/1801.07883**.

ZOU, H. and HASTIE, T. (2005). Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* **67** 301–320.