Winter 12-16-2023

Interpretable Word-Level Sentiment Analysis With Attention-Based Multiple Instance Classification Models

Chenyu Yang
Southern Methodist University, chenyuy@smu.edu

INTERPRETABLE WORD-LEVEL SENTIMENT ANALYSIS

WITH ATTENTION-BASED MULTIPLE INSTANCE CLASSIFICATION MODELS

Approved by:

Dr. Jing Cao
Professor in Department of Statistics
and Data Science, SMU

Dr. Daniel F. Heitjan
Professor in Department of Statistics
and Data Science, SMU

Dr. Lynn Stokes
Professor in Department of Statistics
and Data Science, SMU

Dr. Eric Larson
Professor in Department of Computer
Science, SMU

Dr. Xinlei Wang
Professor in Department of
Mathematics, UTA

INTERPRETABLE WORD-LEVEL SENTIMENT ANALYSIS WITH ATTENTION-BASED MULTIPLE INSTANCE CLASSIFICATION MODELS

A Dissertation Presented to the Graduate Faculty of the

Dedman College

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

with a

Major in Statistical Science

by

Chenyu Yang

Master of Science in Statistics, Rutgers University
MSc in Actuarial Science, Heriot-Watt University, UK
Bachelor's Degree in Bilingual Insurance, SWUFE, China

December 16, 2023

ACKNOWLEDGMENTS

I am deeply grateful to my research advisor, Dr. Jing Cao, for her unwavering guidance and support throughout this journey. Her valuable advice has been instrumental in the completion of this work, and I am truly indebted to her.

I would also like to extend my heartfelt appreciation to my committee members, Dr. Daniel F. Heitjan, Dr. Lynn Stokes, Dr. Eric Larson, and Dr. Xinlei Wang, for their constructive suggestions and advice.

I would also like to express my sincere gratitude to all the faculty members in the department of Statistical Science at SMU. Your expertise, dedication, and passion for research have been a constant source of inspiration for me. Thank you for creating a nurturing and intellectually stimulating environment that has greatly enriched my academic journey. Your support and encouragement have played a significant role in shaping my research and academic growth.

I would like to extend my wholehearted thanks to all my colleagues and friends at SMU. Your friendship and support have been priceless throughout this journey. Your encouragement, camaraderie, and shared experiences have made my time at SMU truly memorable and enjoyable. Thank you for being there for me, for cheering me on, and for making this academic journey all the more rewarding.

Special thanks to Sheila Crain for her exceptional administrative support.

Yang, Chenyu

Master of Science in Statistics, Rutgers University
MSc in Actuarial Science, Heriot-Watt University, UK
Bachelor's Degree in Bilingual Insurance, SWUFE, China

Interpretable Word-Level Sentiment Analysis
With Attention-Based Multiple Instance Classification Models

Advisor: Dr. Jing Cao
Doctor of Philosophy degree conferred December 16, 2023
Dissertation completed August 24, 2023

In this study, our main objective is to tackle the black-box nature of popular machine learning models in sentiment analysis and enhance model interpretability. We aim to gain more insight into the decision-making process of sentiment analysis models, which is often obscure in those complex models. To achieve this goal, we introduce two word-level sentiment analysis models.

The first model is called the attention-based multiple instance classification (AMIC) model. It combines the transparent model structure of multiple instance classification and the self-attention mechanism in deep learning to incorporate the contextual information from documents. As demonstrated by a wine review dataset application, AMIC can achieve state-of-the-art performance compared to a number of machine learning methods, while providing much improved interpretability.

The second model, AMIC 2.0, improves AMIC in two key aspects. Notably, AMIC is limited in integrating positional information in text because it ignores the order of words in documents. AMIC 2.0 comes up with a novel approach to incorporate relative positional information in the self-attention mechanism, enabling the model to capture more accurate sentiment that is position-sensitive. This modification enables the model to better understand how word order and proximity influence sentiment expressions. Secondly, AMIC

2.0 takes a step further by decomposing the sentiment score in AMIC into a context-independent score and a context-dependent score. This decomposition, along with the incorporation of two sentiment shifters linking these scores in a global environment and a local environment of text respectively, elucidate how context of document influences sentiment of words, leading to more interpretable results in sentiment analysis.

The utility of AMIC 2.0 is demonstrated by an application to a Twitter dataset. AMIC 2.0 has improved the overall performance of AMIC, with the additional capability of handling more intricate language subtleties, such as different types of negations. Both AMIC and AMIC 2.0 are trained without having to use pre-trained sentiment word dictionary or seeded sentiment words. Compared to some other big language models, their computation cost is relatively low and they are versatile to use conventional datasets to generate domain-specific sentiment dictionary and provide interpretable sentiment analysis results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

To my loving parents and beloved wife, Your unwavering support, encouragement, and belief in me have been a constant source of inspiration. You have always been there for me, guiding me with love and wisdom. This accomplishment is a reflection of your guidance and sacrifices. Thank you for being my pillars of strength throughout this journey. This work is dedicated to you, with immense gratitude and love.

CHAPTER 1

Sentiment analysis: an introduction

Sentiment analysis (SA) is a field of natural language processing (NLP) that focuses on extracting and understanding subjective information from text. It involves automatically identifying and categorizing sentiment expressed in text, determining whether the sentiment is positive, negative, or neutral.

In today's digital age, with an abundance of user-generated content on social media, online reviews, and customer feedback, SA has gained more popularity with the ever growing collection of text data. It offers valuable insights into public opinion, customer feedback, brand perception, and market trends, etc. Businesses and organizations can leverage SA to monitor online reputation, understand customer sentiment, gauge the success of marketing campaigns, and make data-driven decisions.

SA employ a range of methods, from traditional statistical models to advanced deep learning models. These methods utilize various linguistic and statistical features, including word frequency, word embedding, semantic analysis, sentiment lexicons, and machine learning classifiers, to analyze and classify sentiment expressed in text.

The field of SA has witnessed significant advancements in recent years, with many of the latest breakthroughs stemming from the utilization of neural networks. Neural networks have revolutionized SA by providing powerful tools to tackle the complexities and nuances inherent in text data. These models have shown remarkable success in SA tasks, outperforming many traditional statistical approaches. However, despite the numerous advantages that neural networks brought to SA, one criticism remains to be the

black-box nature of such methods. The term “black box” refers to the challenge of interpreting and understanding the inner workings of neural network models, which in turn obstructs the transparency and clarity of their decision-making processes.

In this dissertation, we propose new methods to conduct word-level context-based SA. Unlike traditional models that produce a single label for an entire piece of text, word-level sentiment analysis focuses on analyzing the sentiment of individual words or short phrases within the text. This approach offers a fine-grained analysis and provides insights into the model’s decision-making process, which helps to reveal the inner workings of the model and yield more interpretable results.

Chapter 2 introduces a word-level contextualized sentiment analysis model, known as the attention-based multiple instance classification (AMIC) model. It combines the transparent model structure of multiple instance classification and the self-attention mechanism in deep learning to incorporate the contextual information from documents. The application to an online wine review dataset demonstrates the model’s capability to conduct fine-grained SA, while attaining the similar state-of-the-art performance compared to a number of machine learning SA models.

Although AMIC can incorporate contextual information which is critical in SA, it ignores the positional information (i.e., the order of words) in text. The positional information can be useful in subtle situations such as negation and presence of intensifier. In Chapter 3, we introduce AMIC 2.0, an upgraded version of AMIC, which incorporates relative positional information in the self-attention mechanism, enabling the model to capture more accurate sentiment that is position-sensitive. It enables the model to better understand how word order and proximity influence sentiment expressions. Another novelty of AMIC 2.0 is that it decomposes the sentiment score in AMIC into a context-independent score and a context-dependent score. This decomposition, along with the incorporation of two sentiment shifters linking these scores in a global environment and a local environment of text respectively, elucidates how the context of a document influences the sentiments of

words, leading to more interpretable results in SA. The utility of AMIC 2.0 is demonstrated by its application to a Twitter dataset that contains 1.6 million tweets.

CHAPTER 2

Attention-based Multiple Instance Classification (AMIC) Model for Word-level Sentiment Analysis

2.1. Introduction

In general, there are three kinds of approaches used in SA: rule-based methods, statistical methods, and neural network methods. Rule-based SA methods typically rely on a sentiment dictionary, which consists of sentiment scores on a collection of words, to calculate the overall sentiment of texts. These methods are known to generalize poorly due to the difficulty in capturing intricacies in language and incorporating domain knowledge and contextual information ([28]).

Statistical SA methods, which employ rigorous probability-based approaches to modeling text data, have a successful track record in handling complex data ([30]). For example, [56] used logistic regression to detect hate speech in tweets. Naive Bayes models and support vector machine have been used to classify movie reviews as positive or negative ([12]). These relatively simple statistical models are often considered interpretable and straightforward, as they assume clear relationships between the input text and the outcome, allowing for a transparent explanation of the model's predictions. However, most of the statistical SA models don't have a mechanism for incorporating document context. They can not deal with situations where words or phrases may carry different sentiments/meanings depending on their contexts, which usually results in less competitive performance.

In recent years, neural network methods in NLP have gained popularity due to their flexible modeling capability. One of the key advantages is their ability to capture document context, where the context of a word or sentence consists of words or sentences before and after it, which helps to elucidate its meaning ([33, 66]). For example, convolutional neural network (CNN) can process image data rich in spatial context, and they can be applied to SA to learn textual features in the same way they learn from image data. [27] first used CNNs for sentiment classification of movie reviews and showed that CNNs can effectively capture local contextual patterns. The long-short term memory (LSTM) model ([20]) and its variant, the Bidirectional LSTM (BiLSTM) model ([65]), were developed to handle sequential data such as texts and time series data. They use recurrent neural network with feedback connections to learn the dependence structure in sequence prediction problems, which naturally incorporates contextual information in SA. One of a more recent developments is the Bidirectional Encoder Representations from Transformers (BERT) model, proposed by [13]. With the self-attention mechanism, it can generate context-aware representations. Its variants, such as RoBERTa ([29]) and DistilBERT ([43]), have achieved excellent results on various benchmark datasets in SA.

Although the neural network models have demonstrated their effectiveness in SA, they are often criticized as being "black box" due to their complex structures and lack of interpretability. Especially for methods which focus on sentence-level or document-level analysis, the reasoning behind the final classification/prediction is often unclear. This lack of interpretability can undermine trust in the results and prevent applications of the models in areas where interpretability is as important as predictive accuracy ([3]). Therefore, there is a growing need for interpretable high-performance SA models that can provide greater transparency in the inference processes.

One way to improve the neural network SA models is to develop an interpretable word-level context-based SA (WCSA) approach. Such a model will provide word-level sentiment measures while incorporating its local and global contexts, giving a more gran-

ular understanding of the key words or phrases that contribute to the overall sentiment of a text. WCSA can be used to create sentiment lexicon, which is a set of words assigned with sentiment scores ([55, 59]). This type of fine-grained analysis is particularly useful in scenarios where the sentiments of individual words is more important than the overall sentiment. For example, in product reviews, the sentiments of individual words may provide more information on the product than the overall sentiment of the review. Consider the following review of an electronic product: "I like my new phone, but I wish the battery life could be improved." The overall sentiment of the review is positive, but when the sentiment is analyzed at word-level, it shows that the reviewer has an issue with the battery life, which can provide useful information for the vendor or manufacturer. This type of analysis can help identify and address specific issues of a product, leading to improved customer satisfaction.

Despite the valuable insights that WCSA can provide, most SA studies do not focus on individual words. Instead, they are conducted at higher levels such as at the sentence level and the document level. This is because assigning sentiment scores to individual words can be challenging due to the ambiguous and context-dependent nature of sentiment. Many words can express different sentiments depending on the context. Taking the word "heavy" for example: it can express a positive sentiment when it is used to describe music (e.g., the bass is heavy and powerful), but it can also express a negative sentiment describing work (e.g., tired from heavy work).

In Chapter 2, we propose a novel model for WCSA which is called the attention-based multiple instance classification (AMIC) model. It has three building components. One is to use word embedding to transform text to data, a technique that maps words to a high-dimensional vector space ([58]). The second one is to conduct WCSA in the framework of multiple instance classification (MIC), which is a weakly supervised learning technique where training instances are arranged in sets (i.e., bags), and label is only provided for the entire bag, as opposed to individual instances ([4]). The third component is to incorporate

the self-attention mechanism which allows the model to capture contextual information by assigning weights to different words in the text based on their relevance to the SA task ([7]). The first two components make the model structure easy to comprehend. The third component inserts contextual information effectively in the model. Combining the three components, AMIC is capable to produce prediction results comparable to the neural network SA models, while providing additional interpretability.

Our work is motivated by a sentiment classification study on wine reviews, which consists of 141,409 reviews collected from the website of the renowned wine magazine *Wine Spectator* dated from 2005 to 2016. Each year, the magazine's editors chose more than 15,000 wines for blind tasting, where they provided tasting notes, numeric ratings, and recommendations. The tasting scores are on a 100-point scale. The majority of wines have a rating in the range of 80–100. [26] applied three neural network methods (i.e., CNN, BiLSTM, and BERD) to classify the wines into different sentiment groups. For example, the sentiment of a wine is labeled as positive if its rating is at least 90, and negative otherwise. The study showed that all three neural network methods produced much more accurate classification than the logistic regression method which only uses numeric variables such as price and age of wine. However, none of the neural network SA methods are able to provide an interpretation of the reasoning behind their classification results.

The aim of our study is to develop an interpretable SA model without compromising its performance compared to the neural network methods. Specifically, we assume that words in a review can be categorized as either sentiment words or function words. Sentiment words are associated with a clear sentiment polarity, describing an emotion or experience that is either pleasant/desirable (e.g., delightful, smooth, pleasant) or unpleasant/undesirable (e.g., sour, unpleasant, corked). On the other hand, function words are words that are used to structure the sentence and convey meaning without sentiment implications, such as most prepositions, conjunctions, articles, pronouns, auxiliary verbs, etc. The proposed AMIC model is able to recognize sentiment words, estimate sentiment

score at the word level, determine the overall sentiment of a review by combining the sentiment scores of individual words, and provide interpretable results in SA.

The chapter is organized as follows. In Section 2.2, we briefly review the previous works on the three components of AMIC. Section 2.3 explains the model structure and the implementation algorithm of AMIC. Section 2.4 includes the application details of AMIC on the wine review data and its comparison to a number of representative SA methods. Section 2.5 concludes with discussion and future directions following this work.

2.2. Related works

2.2.1. Word embedding

The first step in text analysis is to turn text into data. The simplest way to represent a word is to use a one-hot encoded vector which is an index for location in a long vocabulary list. It serves as the foundation of frequency-based encoding methods, which include bag-of-words and term-frequency inverse document frequency (tf-idf) ([46, 50]). Such encoding methods do not provide information about word meaning and they do not reveal any relationships between words.

In AMIC, we use word embedding to represent text as data. Word embedding maps a word in a vocabulary to a latent word representation vector space where words with similar contexts are in proximity. By doing so, a word is converted to a vector that summarises both the word's syntactic and semantic information. Compared to the one-hot encoded vector, word embedding provides us with an efficient, dense representation in which similar words have a similar encoding, where we can use arithmetic operation of vectors to study the relationships of words. For the details of word embedding and the frequency-based encoding methods, readers may refer to [16].

Since the introduction of *word2vec* ([34]) which is a method to efficiently create word embedding, it has quickly become one of the most popular choices to transform text to data. In this study, we will use word embedding as features of words in a multiple instance learning framework to conduct WCSA.

2.2.2. Multiple instance learning

Multiple instance learning (MIL) is a form of weakly supervised learning where the learning task is performed on a set of labeled bags, each containing a collection of instances whose labels are often unobserved. Each individual instance is described by a set of covariates (or features). Instances in a bag contribute to the observed bag-level response (or label). MIL was first introduced by [14] for drug activity prediction. The bag label is positive if at least one instance label is positive, and the bag label is negative if all instance labels are negative. The goal is to predict the label of a new bag. More details of MIL can be found in [4].

When the label is categorical, MIL is referred to as multiple instance classification (MIC). The task of predicting the sentiment of wine reviews can be formulated as an MIC problem. Each wine review can be considered as a bag consisting of individual words as instances, where the features of these instances are represented by the corresponding word embeddings. Predicting the overall sentiment of the reviews is equivalent to predicting the bag labels.

[39] presented an approach based on primary instance, which assumes that the bag label is solely determined by the primary instances, while the non-primary instances carry little information on the bag label. [62] followed this assumption and introduced a Bayesian MIC approach for cancer detection using T-cell receptor sequences. It is composed of two nested probit regression models, where the inner model predicts the primary instances and the outer model predicts bag labels based on the features of the primary instances identified by the inner model. We will implement the idea of identifying primary instances

and the nested regression strategy in MIC to differentiate sentiment words and function words and conduct WCSA in our proposed method.

2.2.3. Self-attention mechanism

Although word embedding, compared to frequency-based encoding methods, can provide a more informative representation of text as data, it is an isolated representation of individual words. In addition, the MIC framework treats instances as independent entities. Both fail to incorporate contextual information in documents. This creates an issue in WCSA, because it implies that a word always receives the same sentiment score, regardless of its context. However, as the earlier example of the word "heavy" demonstrates, the same word may carry very different sentiments in different contexts.

To overcome this limitation, we incorporate the *self-attention* mechanism ([7]) into AMIC. It allows the model to identify which words are more important in relation to other words in the sequence, by assigning different weights to the words based on their relevance to the task at hand. This enables the model to focus on the most important parts of input and better capture the context-dependent nuances of the text. As demonstrated by [13], the incorporation of contextual awareness in BERT leads to significant advancements in the performance of language models, surpassing the previous benchmarks.

The incorporation of the self-attention mechanism allows AMIC to assign context-based sentiment scores to individual words by considering the interactions between words within a document. It facilitates comprehension of subtle sentiment conveyed by words in varying contexts and thereby produces more accurate sentiment scores.

2.3. Methodology

AMIC has a two-layer nested regression structure depicted in Figure 2.1. Layer 1 employs a logistic regression model using the self-attention transformed word embeddings

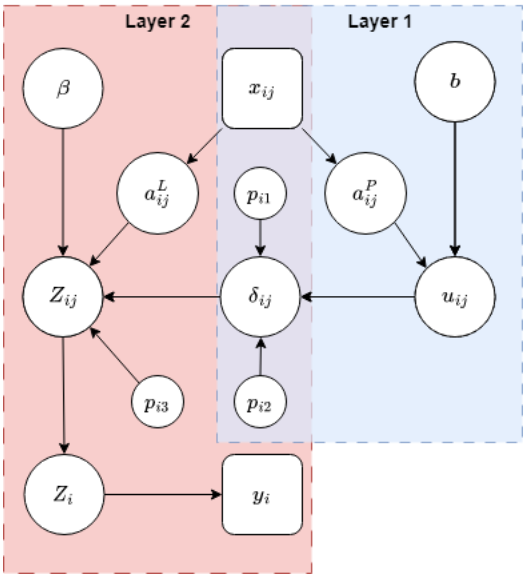

Figure 2.1: Model Structure of AMIC. Layer 1 aims at identifying sentiment words. Layer 2 produces both word-level and document-level context-based sentiment scores.

to predict whether a word is a sentiment word in a document. Layer 2 employs another logistic regression model to estimate word-level context-based sentiment scores of the sentiment words identified in Layer 1, upon which, the classification of the sentiment label for the entire document is obtained.

In the following, we introduce AMIC in the context of binary classification, but it can be easily extended to multiclass classification or prediction with continuous outcomes.

In AMIC, y_i ($i = 1, 2, ..., n$) represents the observed sentiment label of the i^{th} document, x_{ij} is a d-dimension vector (i.e., a length-d word embedding vector) representing the j^{th} ($j = 1, 2, ..., m_i$) word in the i^{th} document. In this study, the documents are padded to have the same length (i.e., $m_i = m$). Note that only x_{ij} and y_i are observed and they are represented with a square box in Figure 2.1.

In Layer 1, the context-independent word embedding x_{ij} is transformed to a context-based word embedding, denoted as a_{ij}^P, using the self-attention mechanism, based on the algorithm described in [57]. Then AMIC conducts a logistic regression using a_{ij}^P as

features of each word to predict δ_{ij}, which is the indicator on whether the j^{th} word in the i^{th} document is a sentiment word. Under this model, b is the logistic regression coefficient vector, and $u_{ij} = a_{ij}^P b$ is the linear predictor.

In Layer 2, x_{ij} is transformed to another context-based word embedding, denoted as a_{ij}^L, using the self-attention mechanism. Note that Layers 1 and 2 each employs their own self-attention transformation, because they serve distinct purposes. Layer 1 focuses on identifying sentiment words, so its attention mechanism is driven to achieve this goal. In contrast, Layer 2 is concerned with estimating sentiment values of the sentiment words identified in Layer 1, thus its attention transformation should focus on sentiment quantification of those words. This differentiation allows the model to concentrate on each task with different focuses, resulting in distinctively transformed embedding vectors. Layer 2 uses a_{ij}^L as predictors to produce the sentiment score estimate, Z_{ij}, for the sentiment words identified from Layer 1. The document-level sentiment score, Z_i, is calculated as the average of Z_{ij}'s in the document, which is fed to a logistic regression supervised by the document sentiment label y_i.

Specifically, the relationships among the entities in AMIC can be summarized as follows. In Layer 1, the goal is to identify sentiment words in the document by using

$$u_{ij} = a_{ij}^P b, \tag{2.1}$$

$$\delta_{ij} = \textit{sig}(u_{ij}), \tag{2.2}$$

where $\textit{sig}(u_{ij}) = \frac{1}{1+e^{-u_{ij}}}$ is the sigmoid function. In Layer 2, the goal is to predict the sentiment label of the document by estimating word-level and document-level sentiment

scores, which are

$$Z_{ij} = \delta_{ij} a_{ij}^{L} \beta, \tag{2.3}$$

$$Z_i = \frac{\sum_{j=1}^{m} Z_{ij}}{m}, \tag{2.4}$$

$$\hat{y}_i = \mathbf{1}_{[0.5,1]}(\textit{sig}(Z_i)), \tag{2.5}$$

where $\hat{y}_i$ is the predicted document label, $\hat{y}_i = 1$ if $\textit{sig}(Z_i) \geq 0.5$ and $\hat{y}_i = 0$ otherwise.

In addition to the entities described above, AMIC includes three penalty terms as follows:

$$p_{1i} = c_1 \sum_{j=1}^{m} (\delta_{ij}(1 - \delta_{ij}))^{\frac{1}{2}}, \tag{2.6}$$

$$p_{2i} = c_2 \sum_{j=1}^{m} \delta_{ij}, \tag{2.7}$$

$$p_{3i} = c_3 \sqrt{\sum_{j=1}^{m} (a_{ij}^{L} \beta)^2}. \tag{2.8}$$

The self-attention mechanism is typically implemented using the gradient descent backpropagation learning algorithm, which calculates the gradients of the loss function to update the model parameters. Note that the indicator function is not differentiable, so we can not use an exact 0/1 indicator in Equation (2.2) for sentiment word identification in Layer 1. Our solution is to use a proxy indicator, δ_{ij}, which is a differentiable sigmoid function, and add a penalty term to ensure that δ_{ij}, after rounding to a certain decimal place, has a dichotomous outcome to adequately approximate a 0/1 indicator function. As shown in Figure 2.2, the function $(\delta_{ij}(1 - \delta_{ij}))^{\frac{1}{2}}$ in penalty p_{1i} has a dome shaped curve. It encourages δ_{ij} to take values close to 0 or 1. The $\frac{1}{2}$ power further helps to create a steep curve near 0 and 1 to drive convergence of δ_{ij} towards 0 or 1. The penalty has a

tuning parameter c_1. With this strategy, the differentiable proxy indicator δ_{ij} will take 0 or 1 after rounding to the 6^{th} decimal place during the model training.

The second penalty term p_{2i} promotes sparsity in the identification of sentiment words. Our preliminary examination of the wine review dataset shows that the sentiment words typically account for less than 30% of all the words in a review. This observation initiates the introduction of sparsity in sentiment word identification in Layer 1. The third penalty term p_{3i} is to ensure the stability in the estimation of Z_{ij} in (2.3). Specifically, it imposes an L2 penalty of $a_{ij}^L\beta$ to prevent the model from arbitrarily inflating $a_{ij}^L\beta$ in situations where δ_{ij} may take close-to-zero values in the early training stage.

The parameters in AMIC are estimated using the gradient descent backpropagation algorithm to minimize the following loss function. It consists of a cross-entropy loss and the penalty terms:

$$L(y_i, Z_i) = -\frac{1}{n}\sum_{i=1}^{n}\left([y_i log(sig(Z_i)) + (1-y_i) log(1-sig(Z_i))] + p_{1i} + p_{2i} + p_{3i}\right). \quad (2.9)$$

By minimizing the loss function in Equation (2.9), AMIC is able to automatically identify sentiment words, produce word-level context-based sentiment scores, predict document-level sentiment label, while doing so without requiring prior information such as seed words (i.e., words whose sentiment polarity or score is given) or a pre-trained sentiment dictionary. Algorithm 1 provides an overview of AMIC's training process. Note that c1, c2, and c3 are tuning parameters that need to be selected to optimize performance of the model. Among them, c2 is primarily responsible for regulating the sparsity of the model. If the value c2 takes is too large, it will result in under-identification of sentiment words, leading to underfitting of the model. Conversely, if c2 takes a value that is too small, it will cause over-identification of sentiment words, causing overfitting of the model. A training-validation-test split of the data is employed to train the model, choose the values of the tuning parameters, and assess the model's predictive performance, respectively.

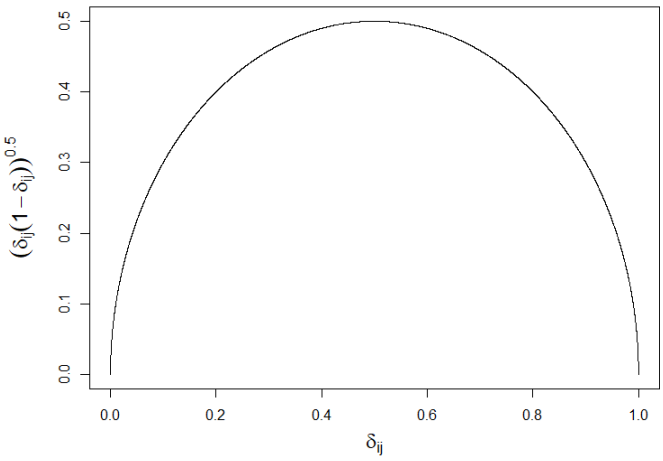

Figure 2.2: Shape of Penalty p_{1i}

Based on the scheme, we choose a value of 1e-3 for c2 which provides the optimal model performance. The classification accuracy is relatively robust to the values of c1 and c3, and they are set to be 1e-4. The training process terminates when the loss evaluated on the validation set stops to be further reduced after 5 consecutive epochs.

2.4. Application: WCSA of wine reviews

2.4.1. Data preprocessing

We have applied AMIC to the same dataset used in [26] and [63], which consists of 141,409 reviews collected from the website of *Wine Spectator* dated from 2005 to 2016. For demonstration purposes, we labeled the sentiment of a wine as positive if its rating is at least 90, and negative otherwise. The reviews typically contain between 20 and 150 words, and we padded short reviews so that all texts have 150 words. The padding was done by inserting a special token, not present in the original vocabulary, with a vector of zero as its embedding. This is to ensure that the padded tokens do not interfere with model training.

Algorithm 1: AMIC training procedure

Data: X_i, y_i
Initialization:
set c1 to 1e-3, c2 and c3 to 1e-4;
while *not stop condition* **do**
    Fix v_{ij} and Calculate δ_{ij} through Layer 1;
    Calculate Z_i through Layer 2 ;
    Evaluate the objective function in (8);
    Update the model parameters (i.e., β, b, and parameters of A^P and A^L) through gradient-descent;
    if *stop condition reached* **then**
        break;
    else
        continue;
    end
end

The word embeddings (i.e., x_{ij}'s in AMIC) are pre-trained results from the Global Vector (GloVe) method ([38]). GloVe learns word embeddings based on co-occurrences of words in a corpus. Several versions of GloVe have been provided based on the corpus used for training. In this project, we used the version of 300-dimensional word embeddings (Glove-300-Wiki) trained on Wikipedia, which is considered to be a reliable approach to provide general-purpose word embeddings.

Words that are not present in the Glove-300-Wiki vocabulary were removed, resulting in an elimination of 3.3$\%$ of the words. Stop words were not removed for two reasons. First, AMIC is able to distinguish between sentiment words and function words. Second, although most stop words do not possess strong sentiment, some of they can still have a noticeable effect modifying sentiment in a text (e.g., "but" or "against"). Thus, removing stop words may result in loss of information in WCSA.

Stemming is a popular preprocessing technique employed in NLP, which is used to reduce a word to its word stem or base word ([60]). In this study, we chose not to apply stemming because it leads to reduced classification accuracy. One possible explanation is that there are words with different forms which carry different sentiment. For example, "acidity" is a word whose noun form has a neutral sentiment in a sentence such as "This

wine has a good level of acidity," but its adjective form "acidic" carries a negative sentiment in a sentence such as "This wine is too acidic and puckers the mouth." In addition, GloVe-300-wiki was trained without stemming, thus applying stemming risks reducing variants of a word to a base form that is not included in the GloVe-300-wiki vocabulary.

Lastly, we eliminated punctuation words and standardized all characters to lowercase. These steps are standard practices in NLP to enhance the robustness and generalizability of the model.

2.4.2. Implementation and comparison

The model was implemented using PyTorch in Python. The training-validation-test split follows a 18:1:1 ratio. The training process terminates when the loss evaluated on the validation set stops to be further reduced after 5 consecutive epochs. The model's prediction accuracy was then reported on the test set using the parameters that produced the lowest loss in the validation set.

To evaluate the performance of AMIC, we compared it to a number of representative SA methods. One of the widely used word-level SA models is the multinomial inverse regression (MNIR) model for text analysis proposed by [53, 54]. MNIR assumes that token count data (i.e., count of words or phrases in text) are drawn from a multinomial distribution. The method uses logistic regression of token counts onto document annotations to obtain low dimension document sentiment representations. Specifically, MNIR is able to create word-level sentiment loadings, which allows for the identification of influential words and phrases in SA. MNIR was implemented using the "textir" package in R. Note that both AMIC and MNIR are word-level SA models. We have also included three neural network SA models: CNN, BILSTM, and BERT, which represent the state-of-the-art SA performance on document-level prediction.

Based on Table 2.1, AMIC has achieved the highest accuracy (0.8926) in the sentiment classification of wine review texts. Among the three neural network models, BERT has

Table 2.1: Model Performances on Sentiment Prediction Accuracy

	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

the highest accuracy at 0.8912, closely followed by BiLSTM and CNN. AMIC, which is developed to conduct interpretable WCSA, does not compromise on its performance in the document-level sentiment classification.

The benchmark word-level MNIR model has achieved an accuracy of 0.8468. The difference in performance between MNIR and the other methods is that MNIR does not have a mechanism to incorporate document context (i.e., the dependence of words in a document). On the other hand, MNIR takes token count as input, which naturally allows n-gram (i.e., continuous sequences of words or symbols in a document) to be the token unit. By comparison, AMIC takes word embedding as input and it can not handle n-gram in the current form. In this study, we also ran MNIR with the bigram input, and its prediction accuracy is 0.8345. It shows that in this particular study bigrams do not provide more information for sentiment classification than individual words.

While AMIC has achieved the best performance in terms of sentiment prediction accuracy for this particular dataset, it is important to recognize that wine review texts are relatively short and straightforward. Hence, it is possible that the neural network models may have better performance on datasets containing longer or more complex texts. Nevertheless, it is reasonable to conclude that AMIC, which has a much simpler model structure, can compete with BERT, the state-of-the-art NLP model, in terms of performance

on certain datasets in SA, with the capability of providing more interpretable results. To further assess the robustness of our findings, we conduct the entire procedure five times using distinct data splits, following the same 18:1:1 ratio. The results of these iterations are presented in Appendix A.1, where the conclusion remains the same as presented above.

As for the computational cost, MNIR is implemented in R and it is trained using a AMD Ryzen 7 3700X CPU. It takes approximately 4 minutes to run with the unigram input, twice as long to run with the bigram input. The other models are implemented in Python and they are trained using a Tesla P100 GPU. The training time for AMIC and CNN is around 20 minutes, and 30 minutes for BiLSTM. BERT, which is a much larger model, takes over 2 hours to train.

2.4.3. Word-level sentiment analysis

AMIC can simultaneously identify individual sentiment words, estimate word-level context-based sentiment scores, and measure the overall sentiment of a document. By doing so, AMIC can identify the relevant parts of the text in determining the overall sentiment and quantify their respective contributions, which provides an interpretable SA framework.

Table 2.2 shows an example of the word-level parameter estimation on a wine review text. Recall that δ_{ij} is the indicator of whether a word is a sentiment word or not. In this review text, words like "this" and "offering" are identified as function words (i.e., $\delta_{ij} = 0$) and their respective sentiment scores (i.e., $\delta_{ij}a_{ij}\beta$'s) are 0. For words that are identified as sentiment words (e.g., "lush," "beauty," etc.), their sentiment scores are aggregated to calculate the document-level sentiment Z_i. This example illustrates how AMIC makes document-level sentiment prediction, which is interpreable from word-level sentiment estimation.

Note that a word appearing in different context may carry different sentiment. AMIC calculates the overall word-level sentiment score for an individual word as the mean of

Table 2.2: Demonstration of AMIC's Interpreability

Raw Text	This lush beauty really tames the rugged structure of the vintage, offering gorgeous boysenberry and raspberry confiture layered with mocha, Turkish coffee, licorice and fig paste notes. The long finish has great grip, though it is remarkably well-embedded, allowing the exotic spice notes to linger beautifully.								
Input Text	this	lush	beauty	...	offering	gorgeous	...	linger	beautifully
δ_{ij}	0	1	1	...	0	1	...	1	1
$\delta_{ij}a_{ij}\beta$	0.0	34.89	29.65	...	0.0	34.18	...	21.04	33.56
Z_i	9.35 (positive prediction)								

its $\delta_{ij}a_{ij}\beta$ values across the documents that it appears in, from which we can create a sentiment dictionary for all the words in the vocabulary. Table 2.3 presents AMIC's top 50 words with the highest sentiment scores in this wine review dataset, where size is proportional to the sentiment score. These words can be roughly grouped into three categories based on their meanings and connotations:

- Descriptive words: words that are used to describe the sensory experience of drinking wine, such as gorgeous, beautiful, ethereal, exquisite, sumptuous, luxuriant, glistening.

- Adjectives of intensity: words that are used to describe the degree or intensity of the sensory experience, such as beautifully, gloriously, gorgeously, thoroughly, amazingly, strikingly, brilliantly, impeccably, perfectly, deliciously, and wonderfully.

- Verbs and nouns: words that are used to indicate the action or the characteristics associated with the wine, such as drip, soak, swirl, stain, sing, silk, perfume, cognac, burgundy, velvet, cascading, haunting, truffle, charms, brunello, fabric, and carpet.

Table 2.3: AMIC'S Top 50 List of Positive Sentiment Words

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

It is less obvious why some of the words in the list convey a positive sentiment compared to the other words. Table 2.4 provides a few examples of how these words, which are not typically associated with positive sentiment in everyday language, convey a positive connotation in the domain of wine reviews. For instance, the adjective "stained" implies a deeper and more complex flavor profile, which is typically considered to be of high quality. Similarly, "carpet" and "fabric" are associated with wines with a rich velvety texture that is often indicative of high quality wines. The word "cascading" introduces an additional flavor profile — suggesting that a wine has a complex and layered flavor profile. Lastly, "cognac" refers to a specific type of brandy, produced in the Cognac region of France, which is renowned for its complex flavor profile and smooth finish, making it a desirable association for high quality wines.

Table 2.5 presents the bottom 50 words with the lowest sentiment scores in this study. They can also be roughly grouped into three categories based on their meaning:

Table 2.4: Illustration of Examples on Less-obvious Positive Sentiment Words

word	review texts
stained	...the long charcoal **stained** finish has a nice tug of roasted bay leaf and truffle, and this shows terrific range...
	...floral notes creamy mouthfeel and a long fruit **stained** finish...
carpet	like a persian **carpet** this lovely elegant champagne seamlessly weaves together its elements with fine grained texture and vibrant acidity...
	...sail across a **carpet** of superfine tannins lingering on the spicy finish...
cascading	...with smoky cherry and red plum flavors **cascading** along a sleek frame a charming wine...
	...and then ends with a **cascading** mix of fruit mineral cedar sage...
fabric	...with fine tannins woven delicately into its **fabric**...
	...like a suit cut from beautiful **fabric** balanced with a vibrancy and silkiness...
cognac	...rich and expressive with spun honey bergamot graphite and pastry dough notes leading to plum tart smoked almond and **cognac** flavors...
	...marzipan fennel seed and **cognac** notes accent this rich and creamy blanc de blancs...

- Adjectives of undesirability: words that in general express undesirability, such as canned, tinny, stale, unfocused, metallic, dull, cloying, muddled, detract, blunt, tired, overripe.

- Adjectives expressing simplicity: words that are used to describe wines that lack depth or complexity: such as quick, generic, simple, canned, uncomplicated, diluted, tinny, neutral, straightforward, flat, fade, modestly, decent, soft, and easy.

- Words related to herbal notes or freshness: words that are associated with fruit or herb implying less desired sense of flavor, such as grassy, weedy, asparagus, parsley, lemonade, chilled, watermelon, greenish, scallion, and cucumber.

Table 2.6 presents a few examples featuring words that are commonly associated with positive sentiment in everyday language but are indicative of negative sentiment in wine reviews. Our analysis of the positive sentiment words reveals that fine wines are generally associated with sophistication, depth, and complexity. Consequently, terms that suggest the opposite qualities, such as “quick” or “breezy,” are deemed undesirable in

Table 2.5: AMIC'S Bottom 50 List of Negative Sentiment Words

quick	generic	hearted	simple	canned
uncomplicated	diluted	tinny	neutral	straightforward
stale	cocktail	easygoing	fizzy	picnic
flat	lovage	greenish	unfocused	breezy
beaujolais	metallic	dull	easy	tail
modestly	decent	fade	scallion	modest
cucumber	cloying	watermelon	soft	parsley
asparagus	kosher	muddled	herbal	lemonade
detract	weedy	blunt	tired	muscadet
grass	grassy	chilled	trim	overripe

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

wine reviews. These terms imply a lack of complexity and evolution. The word "hearted" is associated with the phrase "light-hearted". The expression "light-hearted" is seen as an undesirable attribute, as it suggests a wine that is too simple and lacks character. Finally, terms such as "straightforward" and "easygoing" also indicate a lack of sophistication and depth, making them descriptors for less desirable wines. The respective top 100 and bottom 100 sentiment words produced by AMIC and MNIR are included in Appendix A.2 for further reference.

One group of SA methods are dictionary-based methods where the word-level SA is based on sentiment measures provided by a pre-trained sentiment dictionary. [41] has provided a survey on quantitative tests and qualitative assessments on a number of dictionary-based SA methods. As demonstrated in our study, words used in a particular area may carry sentiment different from its general sentiment. In contrast to dictionary-based methods, AMIC is capable of creating domain-specific sentiment dictionary without referring to an existing one or requiring prior information such as seed words. It enables

Table 2.6: Illustration of Examples on Less-obvious Negative Sentiment Words

word	review texts
quick	...light with modest citrus and green apple hints clean **quick** finish...
	light and **quick** with lemon pulp and jicama notes...
hearted	...a friendly off dry style with light **hearted** candied lime peel and peach notes...
	...light **hearted** and floral with nice melon and kiwi flavors that bounce through...
straightforward	...a **straightforward** white with light notes of gala orange and basil easy to drink but loses some intensity towards the finish...
	..this **straightforward** red shows light cherry herbal and vanilla flavors over ... light tannins
easygoing	...bright cherry and berry notes are up front in this **easygoing** supple red...
	...light and **easygoing** with pretty pear and green melon flavors...
breezy	...tender with modest green apple and green melon notes featuring an open **breezy** finish...
	...pear and floral notes that stay nicely defined through the **breezy** finish...

us to assess sentiment of documents more accurately in the domain of interest, provide more meaningful interpretation, and gain a deeper understanding of the domain-specific wording characteristics.

2.5. Discussion and future work

The main goal of AMIC is to conduct interpretable sentiment analysis without compromising the predictive accuracy. It is achieved by assembling three elements developed in different areas in a novel way. Specifically, it employs word embedding as an efficient transformation of text to data, MIC as an overall transparent model structure, and the self-attention mechanism to incorporate document context in SA. Through this strategy, AMIC combines the best of two worlds: the interpretability of a statistical model and the high predictive performance of deep learning algorithms.

AMIC is not trained using a frequentist or Bayesian approach. Instead, the parameters in AMIC are estimated by minimizing the loss function using the gradient descent back-propagation algorithm. This is because the self-attention transformed word embeddings

need to be updated in each iteration by back propagation, and the associated parameters can not be explicitly expressed in a closed form or in a posterior distribution.

The current AMIC is based on word embedding which is defined on individual words. This restriction makes AMIC only applicable to word-level SA, but not to n-gram or phrase-level SA. Another limitation of AMIC is that it does not incorporate positional information of words within a document. In some cases, the arrangement of words in a sentence can play a crucial role in its sentiment interpretation. For example, in the sentence "The wine is **not** too acidic, the aftertaste is well-balanced", the word "not" negates the negative sentiment of "too acidic", resulting in a positive sentiment. However, if "not" is relocated and the sentence now reads "The wine is too acidic, the aftertaste is **not** well-balanced", it would indicate a negative sentiment. The current version of AMIC is not able to tell the difference in the two arrangements of the same collection of words. Future research will be conducted to address these concerns.

Finally, we provide our thoughts on how AMIC compares to ChatGPT, which is a powerful large language model primarily focused on generating content. ChatGPT provides the unprecedented ability to create human-like text and content (images, music, etc.), and answers questions in a conversational manner. It is certainly more versatile than AMIC in general. We have fed wine review texts to ChatGPT. It can provide a general interpretation, but it tends to classify reviews as positive more frequently and seldom classify a wine view as negative. This is because ChatGPT is not fine-tuned on a specific domain (in this case, the wine review domain), so the output may not be as accurate as a model specifically trained with supervised information in that domain. Taken together, if the research question and research domain is very clear, a specifically trained language model can still be competitive compared to ChatGPT.

CHAPTER 3

AMIC 2.0: Incorporating Positional Information in AMIC for Word-level Sentiment Analysis

3.1. Introduction

AMIC was designed with the objective of providing interpretable results without compromising classification accuracy in SA. To achieve this goal, AMIC incorporates the self-attention mechanism in a multiple instance classification framework, which is shown to be highly effective in conducting contextualized word-level sentiment analysis while delivering impressive classification performance.

The strength of the self-attention mechanism is that it enables the model to assess the relationships between words in text. These connections and relationships between words are commonly referred to as dependencies ([36]). Such dependencies can be categorized into two types: global dependency and local dependency. Global dependency refers to the long-range relationships between words across the entire text. Local dependency focuses on the immediate relationships between neighboring words within a small window of text, which includes dependencies such as subject-verb agreement, adjective-noun relationships, and the impact of negation words on sentiment. Capturing local dependencies can help a SA model to effectively utilize the subtle nuances and relationships among words that collectively contribute to the overall sentiment expressed in text.

The self-attention mechanism was originally designed to emphasize the relationships between words throughout the entire sentence, regardless of their specific positions. This

characteristic makes it highly proficient in capturing global dependencies. However, its ignorance to the positions of words introduces a limitation when it comes to incorporating local dependencies. Without the knowledge of neighboring words, the self-attention mechanism, on its own, struggles to accurately discern sentiment shifts influenced by factors such as negation, word order, or proximity to other words. These contextual cues can be important in determining the sentiment expressed in text, and the self-attention mechanism may not fully grasp these subtleties.

Let's consider the following two sentences as an example, where the only difference is the placement of the word "not":

> Sentence I: The service of the restaurant is good, the overall experience is **not** bad.
> Sentence II: The service of the restaurant is **not** good, the overall experience is bad.

Sentence I presents a positive assessment of a restaurant, highlighting its excellent service. In contrast, Sentence II expresses a negative sentiment towards the overall dining experience as well as the quality of service provided. This simple example illustrates the importance of positional information in understanding the meaning of text. By rearranging the position of just one word, the entire sentiment of the text is reversed. Note that due to its utilization of the self-attention mechanism, AMIC treats the collection of words in text as an unordered set and ignores their positional information. Thus, AMIC would produce identical sentiment estimation for Sentence I and Sentence II, resulting in the failure to differentiate between them.

In this chapter, we propose AMIC 2.0 to overcome the lack of positional awareness in AMIC. Our approach is based on the methodology proposed by [47], which incorporates relative positional information into the self-attention mechanism. By adding the positional information into the self-attention process, AMIC 2.0 can take into account the appro-

priate word order and contextual dependencies, which makes it capable of conducting WCSA more accurately than AMIC, capturing the local dependencies between words, understanding the sentiment nuances, and providing more interpretable results in SA.

Another improvement in AMIC 2.0 is the enhanced model structure over AMIC. AMIC incorporates the multiple instance classification framework, which improves SA interpretability by explicitly distinguishing sentiment words versus function words, producing contextualized sentiment scores for sentiment words, and generating a document-level sentiment score as the average of word-level sentiment scores. To understand how context influences sentiment expressed in text, AMIC 2.0 takes a step further by decomposing the contextualized sentiment score in AMIC into two distinct scores: the context-independent and context-dependent scores. The context-independent sentiment score serves as a baseline sentiment estimation that remains consistent regardless of the surrounding context, while the context-dependent score is affected by both the global dependency and the local dependency. By providing estimations for the two sentiment scores and quantifying the global dependency and the local dependency respectively, AMIC 2.0 can offer insights on how context influences sentiment and the inner workings in the model's decision-making process.

To demonstrate the model's capability in capturing positional relationships that influence the polarity and intensity of sentiment words, it requires to use datasets that contain abundant examples of these positional structures to train the model. In this study we use the Sentiment140 dataset introduced in [18]. The Sentiment140 dataset is sourced from Twitter, a popular social media platform where users share brief messages called tweets. This dataset consists of 1.6 million tweets collected between April 6, 2009, and June 25, 2009. Among these tweets, 800,000 are labeled with negative sentiment, while the remaining 800,000 are labeled with positive sentiment. These tweets exhibit diverse linguistic variations that are context-sensitive. Therefore, it is critical for the model to un-

derstand the relative positional information in tweets to accurately predict the sentiment expressed in each message.

The subsequent sections of this chapter are structured as follows. In Section 3.2, we provide a detailed overview of prior research work on the critical components in AMIC 2.0. Section 3.3 explains the model's construction and the implementation algorithm, outlining the key technical components of AMIC 2.0. Section 3.4 includes the application details of AMIC 2.0 on the Sentiment140 dataset. Section 3.5 summarizes the model's contributions and presents discussion on potential future research directions.

3.2. Background

3.2.1. A closer look at the self-attention mechanism

Because this project involves modifying the self-attention mechanism, as opposed to just applying it as in AMIC, here we provide a detailed examination of how self-attention operates at its core. As discussed before, the motivation behind self-attention is to enable neural networks to capture the relationships between words in a sequence more effectively.

Traditional neural network architectures such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs) have limited ability to capture relationships between words, particularly long-term dependencies. For example, RNNs suffer from the vanishing gradient problem, which prevents them from effectively capturing long-term dependencies. CNNs only analyze the relationships between words within a fixed window (also known as a "kernel"), ignoring long-term dependencies. This can limit their ability to capture the complex relationships between words in a sequence, particularly when the dependencies between the words span a longer distance.

On the other hand, the self-attention mechanism enables neural networks to capture global dependencies by computing a weighted sum of the input words, where the weights are computed based on their relationships with all other words in the sequence indiscriminatel. This enables the network to focus on different parts of the input sequence depending on their relevance to the current word and capture complex patterns and relationships between the words, regardless of their distance from the current word. To achieve this, the self-attention mechanism follows three major steps.

The initial step involves computing Query, Key, and Value vectors. Each word in the input sequence is represented as a vector, and the self-attention mechanism computes separate Query, Key, and Value vectors through linear transformations. This concept of Key, Value, and Query is analogous to information retrieval systems in search engines. For instance, when a user search for videos on Youtube, the search engine will map the user's query (text in the search bar) against a set of keys (video title, description, genre, etc.) associated with candidate videos in its database, and present the user with retrieved values (i.e., the best-matched videos).

Similarly, in the self-attention mechanism, Query vectors represent the words that require attention, Key vectors represent the words used to calculate attention weights, and Value vectors represent the information that the network aims to retrieve and propagate to the next layer of the neural network. By utilizing these vectors, the self-attention mechanism effectively captures and leverages the relationships between words in the input sequence.

The widely utilized self-attention mechanism is known as the scaled dot-product attention. Consider an input sequence of word embedding vectors for a document denoted by $x = [x_1, x_2, ..., x_m]$, where $x_j \in \mathbb{R}^d$ represents the embedding for the j^{th} word and m is the total number of words in the sequence. Self-attention uses three $d \times d$ projection matrices W^Q, W^K, and W^V, which are updated as model parameters during training. These matrices serve to project the inputs into Query, Key, and Value vectors for each word,

respectively, via matrix multiplication between the matrices W and the embedded inputs x_j,

$$\text{Query vector} : x_j^Q = x_j W^Q$$
$$\text{Key vector} : x_j^K = x_j W^K$$
$$\text{Value vector} : x_j^V = x_j W^V.$$

Next the self-attention mechanism proceeds to compute a weight vector for each word, considering its interactions with other words in the sequence. Each element in the attention weight vector is calculated by evaluating the strength of relationship between two words through an inner product. For instance, in the self-attention representation of word j, the similarity between word j and word k is computed as follows:

$$e_{jk} = (x_j^Q)(x_k^K)^T \tag{3.1}$$

The attention weight α_{jk} is then computed as the standardized similarity score of e_{jk}:

$$\alpha_{jk} = \frac{\textit{exp}(e_{jk})}{\sum_{h=1}^{m} \textit{exp}(e_{jh})}, \tag{3.2}$$

where α_{jk} can be interpreted as the level of attention that should be given to the k^{th} word when calculating the context-dependent representation for the j^{th} word.

As an illustration, let's take the sentence "the food was so bad" as an example. In this sentence, the word "bad" is crucial for determining the sentiment, while the word "so" intensifies the negative sentiment conveyed by "bad". The self-attention mechanism takes into account the relationships between the words to determine how much attention should be paid to each word. Therefore, when computing the attention weights for the word "bad", the mechanism would assign a higher weight to the word "so" because it

intensifies the sentiment of “bad”, while assigning a lower weight to the word “the”, as it has little influence on “bad”.

In the third and final step, the self-attention mechanism computes a weighted sum of the Value vectors for each word in the input sequence. The weights, calculated in the second step, measure the relevance of each word in the input sequence to the current word. The self-attention embedding of word j, a_j, which incorporates the context of word j, is computed as follows,

$$a_j = \sum_{k=1}^{m} \alpha_{jk}(x_k^V) \tag{3.3}$$

The Value vectors x_k^V $(k = 1, ...m)$ are commonly regarded as encodings of the semantic meaning of words in the input sequence. They capture the fundamental semantic information that plays a crucial role in comprehending the entire sequence, thus proving valuable for various downstream tasks like sentiment analysis ([52]) machine translation ([17]), or question-answering ([45]). In Equation (B.3), a_j not only includes the semantic encoding of word j itself but also incorporates the information from all the other words weighted by their attention relevance.

3.2.2. Incorporation of relative positional information

The self-attention mechanism is a powerful technique in NLP that makes use of context by computing attention weights for word representations. However, it relies exclusively on the inner product computation between the Query and Key vectors to calculate the attention weights. As a result, the self-attention mechanism does not inherently incorporate the positional information of words within the input sequence.

To illustrate this point, let’s refer to the example of Sentence I and II discussed in Section 3.1, where the position of one word is crucial for correctly interpreting the meaning of the sentences. The self-attention mechanism ignores the positional information of words and treats the two sentences in the same way. This limitation hinders the model’s abil-

ity to capture the nuanced meaning conveyed by the word order, leading to inaccurate interpretation of the underlying sentiment.

To tackle the challenge of incorporating positional information into the self-attention mechanism, researchers have proposed different approaches ([57], [47],[2], [6] and [22]). One method, introduced by [47], is known as self-attention with relative positional representation, where relative positional information refers to the information about the relative distances or positions between the words in an input sequence. This technique enables the self-attention mechanism to effectively utilize the relative positional information between words which can convey important information about the relationships between the words and their syntactic or semantic association.

For instance, in the sentence "I do not like cooking," the relative position of the words play a crucial role in interpreting the sentiments of both individual words and the overall sentence. The word "not" functions as a negation word, reversing the polarity of the sentiment that follows it. Specifically, the word "not" negates the positive sentiment that would be conveyed by the word "like" if it were used independently. Consequently, the phrase "do not like" conveys a negative sentiment, even though the individual words "do," "not," and "like" might be positive or neutral. To accurately understand the sentiment of the phrase, an SA model needs to capture the relative positional information between the words "not" and "like." This enables the model to recognize the negation and correctly assign a negative sentiment to the sentence.

The approach introduced in [47] aims to explicitly capture the relative positional information within an input sequence of words. This is achieved by introducing relative positional embeddings, which encode both the position and direction between words. For convenience of explanation, we will designate one word as the "target word" and the other as the "reference word" within a pairwise relationship.

In the above example, if we set “not” as the target word, the remaining words are considered as reference words. For instance, the reference word “do” is one position before the target word “not”, while the reference word “cooking” is two positions after “not”. In [47], these relative positional relationships are explicitly captured using vectors known as positional embeddings. This approach is similar to how word embeddings represent the meaning of words.

To be easily integrated in the self-attention mechanism, the relative positional embeddings w are constructed as the same d-length vectors as the word embedding vectors. For example, in the left-to-right direction, the relative position between a target word and the reference word one position to the right is represented as the w_1 vector, while the relative position between the target word and the reference word two positions to the right is denoted as w_2, and so forth. In the opposite direction, the relative position between a target word and the reference word immediately to its left is represented as w_{-1}, and the embedding between a target word and the reference word two positions to the left is denoted as w_{-2}, and so forth. Moreover, w_0 denotes the relative position between the target word and itself as the reference word. Given an input sequence of five words, there are in total nine distinct relative positional relationships, represented as w_{-4}, w_{-3}, $\ldots$, w_0, $\ldots$, w_3, and w_4, as illustrated in Figure 3.1.

Given that these relative embedding vectors represent the distance between two words, we can also express them using a pairwise notation, which is denoted as $a_{j \to k}$, where j represents the position of the target word and k represents the position of the reference word. Furthermore, [47] considers position of words only up to a specific limit, which is based on the assumption that the relative position of two words carries trivial information beyond a certain threshold. Let parameter q denote the relative position threshold, the model learns position embeddings w_{-q}, w_{-q+1}, $\ldots$, w_{q-1}, and w_q, resulting in a total of $2q + 1$ unique relative positional embeddings. Figure 3.2 illustrates this arrangement with $q = 2$.

Target word j / Reference word k	I $j=0$	do $j=1$	not $j=2$	like $j=3$	cooking $j=4$
I $k=0$	$w_0 = a_{0\to0}$	$w_{-1} = a_{1\to0}$	$w_{-2} = a_{2\to0}$	$w_{-3} = a_{3\to0}$	$w_{-4} = a_{4\to0}$
do $k=1$	$w_1 = a_{0\to1}$	$w_0 = a_{1\to1}$	$w_{-1} = a_{2\to1}$	$w_{-2} = a_{3\to1}$	$w_{-3} = a_{4\to1}$
not $k=2$	$w_2 = a_{0\to2}$	$w_1 = a_{1\to2}$	$w_0 = a_{2\to2}$	$w_{-1} = a_{3\to2}$	$w_{-2} = a_{4\to2}$
like $k=3$	$w_3 = a_{0\to3}$	$w_2 = a_{1\to3}$	$w_1 = a_{2\to3}$	$w_0 = a_{3\to3}$	$w_{-1} = a_{4\to3}$
cooking $k=4$	$w_4 = a_{0\to4}$	$w_3 = a_{1\to4}$	$w_2 = a_{2\to4}$	$w_1 = a_{3\to4}$	$w_0 = a_{4\to4}$

Figure 3.1: The target words are listed in the first row, and the reference words are listed in the first column. The positional relationship between a reference word and a target word is represented as the w's.

k \ j	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$
$k=0$	$w_0 = a_{0\to0}$	$w_{-1} = a_{1\to0}$	$w_{-2} = a_{2\to0}$	$w_{-2} = a_{3\to0}$	$w_{-2} = a_{4\to0}$
$k=1$	$w_1 = a_{0\to1}$	$w_0 = a_{1\to1}$	$w_{-1} = a_{2\to1}$	$w_{-2} = a_{3\to1}$	$w_{-2} = a_{4\to1}$
$k=2$	$w_2 = a_{0\to2}$	$w_1 = a_{1\to2}$	$w_0 = a_{2\to2}$	$w_{-1} = a_{3\to2}$	$w_{-2} = a_{4\to2}$
$k=3$	$w_2 = a_{0\to3}$	$w_2 = a_{1\to3}$	$w_1 = a_{2\to3}$	$w_0 = a_{3\to3}$	$w_{-1} = a_{4\to3}$
$k=4$	$w_2 = a_{0\to4}$	$w_2 = a_{1\to4}$	$w_2 = a_{2\to4}$	$w_1 = a_{3\to4}$	$w_0 = a_{4\to4}$

Figure 3.2: When the range is limited to $q = 2$, only five distinct positional embeddings are learned, namely w_{-2}, w_{-1}, w_{-0}, w_1, and w_2. The relative positions are considered the same for the cells marked in grey.

To incorporate the relative positional information in the self-attention mechanism, two separate sets of relative positional embeddings are learned in the Key vectors and Value vectors, respectively. Specifically, one set of relative position embeddings, denoted as $a^K_{j\to k}$, is added to the Key vector x^K_j in Equation (3.1) to inject the relative positional information into the Key vector as demonstrated in Equation (3.4). Thus, $(x^K_k + a^K_{j\to k})$ contains both the semantic information and the positional information. A similar operation is conducted by adding another set of relative position embeddings, denoted as $a^V_{j\to k}$, to the Value vector x^V_j. The incorporation of relative positional information in the Key vectors and Value vectors, modifying Equation (3.1) to Equation (3.4) and Equation (B.3) to Equation (3.6), enables the self-attention mechanism to consider the positional relationships be-

tween words and incorporate them in the computation of attention weights.

$$e_{jk} = (x_j^Q)(x_k^K + a_{j\to k}^K)^T \tag{3.4}$$

$$\alpha_{jk} = \frac{\exp(e_{jk})}{\sum_{k=1}^{m} \exp(e_{jk})} \tag{3.5}$$

$$a_j = \sum_{k=1}^{m} \alpha_{jk}(x_k^V + a_{j\to k}^V) \tag{3.6}$$

Generally speaking, in the self-attention mechanism, the query is the information that is being looked for, the key is the context or reference, and the value is the content that is being searched. Thus, Key vectors and Value vectors contain different information of words, and their encoding of relative positional information can be different. We need to have two sets of positional embeddings (i.e., $a_{j\to k}^K$ and $a_{j\to k}^V$), each of them is tailored to produce the most relevant positional information to the respective vector. Together, they ensure an effective incorporation of positional information in the self-attention mechanism.

By introducing these two sets of relative positional embeddings, the model gains an awareness of the neighbouring words to each target word and can potentially assign more significance to these neighboring words within a specific range parameterized by q. This capability enables the model to better understand the context and positional relationships between words, leading to more accurate and contextually informed representation in the self-attention process.

3.2.3. Ngation handling in NLP

Negation is a linguistic phenomenon that reverses the meaning of a word or a sentence. Negation handling refers to methods of automatically detecting the extent of negation and inverting the polarity of opinionated words that are impacted by a negation. It is an important sub-task in sentiment analysis in NLP which is considered as one of the most challenging problems in NLP in opinion mining ([48]).

Early research on negation handling mainly used rule-based approaches applied in the medical domain, where negation handling techniques were primarily employed to analyze medical reports and identify the presence or absence of specific disease conditions. For example, a typical task might involve automatically detecting diseases in medical reports like the following:

> The patient exhibits high fever, but no headache...

where “fever” is a medical condition that requires recognition, while “headache” is considered a medical condition that is not present as it is ruled out by the negation word “not”. In the context of negation handling, the words responsible for negation are termed “cue,” and the portion of the sentence influenced by the cue is referred to as the “scope.” In this example, the cue is the word “not” and the scope of negation is the range within the two words to the right of the cue.

In the medical domain, rule-based approaches for negation handling gained popularity due to the available set of predefined negations and the relatively standardized language used in medical reports ([48]). For instance, when processing a medical report, the decision-making process of a simplified rule-based negation handling system could be:

- Step 1. Retrieve negation terms from a pre-defined negation dictionary.
- Step 2. Search through the report for medical conditions and their positional information with respect to the nearest negation terms.
- Step 3. Report the medical conditions outside the negation scope as present and those within the negation scope as absent.

In practice, rule-based systems often utilize a combination of regular expressions ([1]), sentence parsers which is a NLP tool that analyzes and deconstructs the grammatical

structure of a sentence ([25]), and other methods to facilitate the identification of negation cues and their scope. For example, [5] introduced the NegEx algorithm, a rule-based method that identifies the scope of negation words in discharge reports to determine whether the disease condition is negated or not. DEEPEN, proposed in [31] , is a more advanced negation algorithm that considers the dependency relationship between negation words and concepts. DEEPEN incorporates the Stanford dependency parser to enhance negation detection in medical reports. Similarly, [49] employed a dependency parser for negation detection in clinical datasets, aiming to improve the performance of cTAKE (Clinical Text Analysis and Knowledge Extraction System) which is a modular system of pipelined components combining rule-based and machine learning techniques aiming at information extraction from the clinical narrative ([44]). NegFinder, proposed in [35], introduces a set of rules to recognize a broad range of negated patterns in text by utilizing regular expressions based on a subset of context-free grammars, known as LALR(1) grammars ([15]). NegFinder is trained using a diverse collection of manually inspected and labeled medical documents from a variety of disease conditions.

The majority of these methods depend on annotated datasets for model implementation, leading to the creation of several negation-annotated datasets in various domains. For instance, [51] introduced the BioScope corpus, a valuable resource for studying negation in medical texts. The corpus comprises medical texts, biological papers, and biological scientific abstracts. Expert linguists meticulously annotated the dataset at the token level, identifying negative and speculative keywords, as well as at the sentence level, determining their linguistic scope.

Likewise, in the SA field, most early research for negation handling used rule-based methods focusing on identifying scope and cue of negation in text ([24], [37], [19], [21], and [64]). For instance, [24] studied the influence of the negation term “not’ ’ using dependency parsers along with static and dynamic delimiters. The researchers introduced

heuristic rules that involved sentimental verbs, sentimental adjectives, and sentimental nouns, as well as double object rules for scope detection.

However, in domains that employ non-standardized language, such as online customer reviews or social media platforms like Twitter, coming up with explicit rules to handle negation exhaustively can be challenging. In such cases, it is argued that statistical approaches including machine learning outperform rule-based systems ([11]). For instance, [10] applied support vector machine to the Simon Fraser review corpus, which is a collection of movie, book, and consumer product reviews with annotated negation, and it was found to outperform rule-based approaches in terms of scope detection. In [42], a type of graphic machine learning model called conditional random field ([9]) was used to conduct negation scope detection on a negation-annotated Twitter dataset by exploiting valence shifters, which are changes in the emotional polarity or sentiment of words or expressions in the tweets.

A major limitation of these approaches is their dependence on annotated datasets that usually need to be manually labeled ([23]). The limited availability of negation-annotated datasets poses a significant obstacle, making it difficult to apply these models in various domains. Thus, it is meaningful to develop models which are capable of capturing negation without reliance on annotated datasets.

3.3. Methodology

Two features introduced in AMIC 2.0 distinguishes it from AMIC. First, AMIC 2.0 decomposes the word-level contextualized sentiment score in AMIC into a context-independent score, which represents a word's intrinsic context-free sentiment, and a context-dependent score, which captures the sentiment in the context of a given text. Furthermore, the context-dependent score is obtained by considering both the global dependency and the local dependency, which helps to elucidate how context affects sentiment in differ-

ent angles. Second, AMIC 2.0 implements a modified self-attention mechanism which incorporates the relative positional information of words to quantify the local dependency, enabling it to handle more delicate lingistic complexities such as negation and sentiment intensifier. Last but not least, the training of AMIC 2.0 does not require special annotated datasets to handle lingistic complexity.

Figure 3.3 presents the overview of AMIC 2.0 model structure. Following the notations assumed in AMIC, we use x_{ij} to denote the input word embedding for the j^{th} word and y_i to represent the sentiment label in document i. Furthermore, v_{ij} is a real-valued scalar that represents the context-independent sentiment score of word j in document i. This value captures the inherent sentiment of the word, where a positive value indicates positive sentiment and a negative value negative sentiment.

The newly added r^g_{ij}, r^l_{ij}, and r^s_{ij} in AMIC 2.0 are the context-dependent d-dimensional vectors for the input word x_{ij}. Specifically, r^g_{ij} contains information on its global dependency, r^l_{ij} on its local dependency, and r^s_{ij} contains information to decide whether the word is a sentiment word. Note that r^g_{ij} and r^s_{ij} are computed using the standard self-attention mechanisms without incorporating positional information, whereas r^l_{ij} is computed using the modified self-attention to incorporate relative positional information.

Furthermore, δ^g_{ij}, δ^l_{ij}, and δ^s_{ij} are real-valued scalars derived from the d-dimensional vectors r^g_{ij}, r^l_{ij}, and r^s_{ij}. Specifically, δ^g_{ij} and δ^l_{ij} are termed as global and local sentiment shifters, respectively. The value of δ^g_{ij} ranges from 0 to 10, quantifying the influence of global dependency for the sentiment of the j^{th} word in document i. Whereas δ^l_{ij} represents the impact of local dependency on the sentiment of word j and it takes a value between -1 and 1. A negative value of δ^l_{ij} indicates negation of the context-independent sentiment of word j in document i. Finally, δ^s_{ij} is the proxy indicator that takes a value of either 1 or 0, where 1 indicates that the word j is considered to be a sentiment word and 0 a function word.

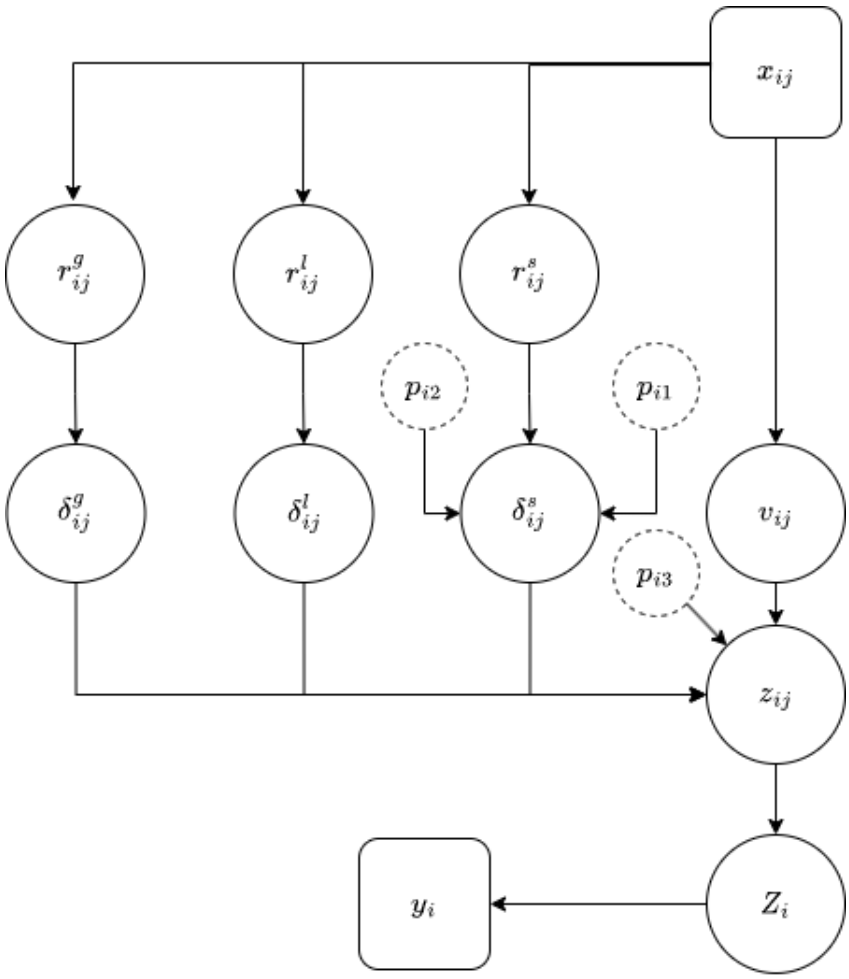

Figure 3.3: Model structure of AMIC 2.0

The rest of the parameters have the same meaning/utility as those used in AMIC. Specifically, p_{i1}, p_{i2}, and p_{i3} are the penalty terms, z_{ij} is the context-dependent sentiment of the j^{th} word in the i^{th} document, and Z_i represents the overall sentiment score of document i. In the following, we will explain the details in the construction of the AMIC 2.0 model.

Let $X_i = \begin{bmatrix} x_{i1} & , \ldots, x_{im} \end{bmatrix}$, a $d \times m$ matrix, represent the sequence of all the input word embeddings for document i. In the initial step, we use a feedforward layer to calculate v_{ij}, the context-independent sentiment score, where b^v is the d-dimensional coefficient vector:

$$v_{ij} = x_{ij} b^v \tag{3.7}$$

Because x_{ij} is pre-trained word embedding, it does not incorporate context of the given text. As demonstrated in Equation (3.7), v_{ij} is solely dependent on its corresponding word embedding x_{ij} and the regression coefficient b^v, without being influenced by the other word embeddings x_{ik} ($k \neq j$). Thus v_{ij} is context-independent, as it is not influenced by other words in the document.

Next we compute the collection of r^g_{ij} and r^s_{ij}. Let $R^s_i = \begin{bmatrix} r^s_{i1} & \ldots & r^s_{im} \end{bmatrix}$ and $R^g_i = \begin{bmatrix} r^g_{i1} & \ldots & r^g_{im} \end{bmatrix}$. Since R^g_i contains the word representations of global dependency and R^s_i is used to identify sentiment words, they are not influenced by the order of words. Thus, both R^g_i and R^s_i are calculated using the original self-attention mechanism:

$$R^g_i = F^g_{\textit{self-attention}}(X_i) \tag{3.8}$$

$$R^s_i = F^s_{\textit{self-attention}}(X_i) \tag{3.9}$$

where $F^g_{\textit{self-attention}}$ and $F^s_{\textit{self-attention}}$ denote the original self-attention algorithm with the details covered in Appendix B.1.

To measure the local dependency and understand the relationships between neighboring words in a document, we have introduced r_{ij}^l, the word representation for the local dependency. In AMIC 2.0, we propose a novel variation of the self-attention mechanism to compute r_{ij}^l which effectively incorporates relative positional information.

We first calculate the Query vector x_{ij}^Q and Key vector x_{ij}^K using the self-attention mechanism. Notice that in comparison to the method listed in Section 3.2.1, we have introduced an additional subscript i to indicate the document index i.

$$\text{Query vector} : x_{ij}^Q = x_{ij} W^Q \tag{3.10}$$

$$\text{Key vector} : x_{ij}^K = x_{ij} W^K. \tag{3.11}$$

We do not compute the Value vector x_{ij}^V since its inclusion is unnecessary in our modified approach for reasons explained later.

Based on [47] in Section 3.2.2, the context-based word embedding a_j is computed following Equation (3.4), (3.5), and (3.6). In our proposed algorithm, the steps are:

$$e_{ijk} = (x_{ij}^Q)(x_{ik}^K + a_{j\to k}^K)^T \tag{3.12}$$

$$\alpha_{ijk} = \frac{\exp(e_{ijk})}{\sum_{h=1}^{m} \exp(e_{ijh})} \tag{3.13}$$

$$r_{ij}^l = \sum_{k=1}^{m} \alpha_{ijk}(a_{j\to k}^V) \tag{3.14}$$

where Equation (3.12) and (3.13) are designed similarly to Expression(3.5) and (3.6).

The difference lies in Equation (3.14) where we modify Equation (3.6) by eliminating the Value vector x_{ij}^V from the word representation calculation. If we follow Equation (3.6), which is $r_{ij}^l = \sum_{k=1}^{m} \alpha_{ijk}(x_{ij}^V + a_{j\to k}^V)$, then r_{ij}^l would be a weighted representation of the Value vectors modified by the positional information. Following Equation (3.14), r_{ij}^l now is a weighted representation of the positional embedding. This adjustment allows r_{ij}^l to

focus on the positional information rather than encoding the semantic meaning carried by the Value vectors. By removing the Value vector of words, the model becomes more sensitive to the relative positions of words, enabling it to better capture the contextual dependencies based on their positions in the sequence. The reason why AMIC 2.0 can let r_{ij}^l only focus on the positional embedding is that AMIC 2.0 has decomposed the semantic/sentiment information of words in multiple latent dimensions: context-independent score and context-dependent score, where the transition from the former to the latter is further explained by the joint function of global shifter and local shifter, so that r_{ij}^l does not have to be a comprehensive word representation but a specific word representation focusing on local positional dependency.

In the following step, δ_{ij}^g, δ_{ij}^l, and δ_{ij}^s are obtained using the update on r_{ij}^g, r_{ij}^l, and r_{ij}^s. Specifically,

$$\delta_{ij}^s = \textit{sig}({r_{ij}^s}^T b^s) \tag{3.15}$$

$$\delta_{ij}^g = 10 \times \textit{sig}({r_{ij}^g}^T b^g) \tag{3.16}$$

$$\delta_{ij}^l = \textit{tanh}({r_{ij}^l}^T b^l). \tag{3.17}$$

Note that δ_{ij}^s eventually acts as an 0/1 indicator function with the constraints added by the penalty terms (see the details in the construction of AMIC). A scaling factor of 10 is applied to the sigmoid function for the global sentiment shifter δ_{ij}^g to address the potential vanishing gradient problem. The hyperbolic tangent function is used to compute δ_{ij}^l which has an output range between -1 and 1, allowing for the detection of sentiment negation. Specifically, a negative value of δ_{ij}^g indicates a flipped sentiment, signifying the presence of sentiment negation.

The penalty terms are constructed in a similar fashion as in AMIC:

$$p_{i1} = c_1 \sum_{j=1}^{m} (\delta_{ij}^s (1 - \delta_{ij}^s))^{\frac{1}{2}}, \tag{3.18}$$

$$p_{i2} = c_2 \sum_{j=1}^{m} \delta_{ij}^s, \tag{3.19}$$

$$p_{i3} = c_3 \sqrt{\sum_{j=1}^{m} (v_{ij} \times \delta_{ij}^g \times \delta_{ij}^l)^2}. \tag{3.20}$$

The utility of the penalty terms and the choice of the tuning parameters remain the same as in AMIC.

The context-dependent sentiment score of word j, denoted as z_{ij}, is obtained by multiplying together the context-independent sentiment v_{ij}, the sentiment word indicator δ_{ij}^s, the global sentiment shifter δ_{ij}^g, and the local sentiment shifter δ_{ij}^l:

$$z_{ij} = v_{ij} \times \delta_{ij}^s \times \delta_{ij}^g \times \delta_{ij}^l. \tag{3.21}$$

The product of these factors captures their collective impact on the final sentiment score for word j. Recall that if a word is identified as a function word (i.e., $r_{ij}^s = 0$), its context-dependent sentiment score is 0 and it does not contribute to the calculation of the document-level sentiment score Z_i. For a sentiment word (i.e., $r_{ij}^s = 1$), the context-dependent score z_{ij} stems from the context-dependent score v_{ij} modified by its global dependency and its local dependency to its context in the document. The document-level sentiment score Z_i is then determined by averaging the sentiment scores of the sentiment words within the document:

$$Z_i = \frac{\sum_{j=1}^{m} Z_{ij}}{\sum_{j=1}^{m} r_{ij}^s}, \tag{3.22}$$

$$\hat{y}_i = \mathbf{1}_{[0.5,1]}(\textit{sig}(Z_i)), \tag{3.23}$$

where $\hat{y}_i$ denotes the predicted sentiment label. $\hat{y}_i = 1$ if $\mathit{sig}(Z_i) \geq 0.5$ and $\hat{y}_i = 0$ otherwise.

Now with the introduction of Equation (3.21), we can add more explanation to the scaling factor 10 in (3.16). The choice of 10 as the scaling factor is somewhat arbitrary but serves the purpose of preventing the product of the global sentiment shifter and the local sentiment shifter from becoming too small. When two values with absolute values smaller than 1 are multiplied together, the result becomes even smaller, potentially leading to vanishing gradients and slower learning during training. By using a scaling factor of 10, the product remains in a reasonable range, helping to keep a more stable training process.

Like AMIC, the parameters in AMIC 2.0 are estimated using the gradient descent algorithm to minimize the following loss function, which consists of a cross-entropy loss and the three penalty terms:

$$l(X_i, y_i) = -\frac{1}{n}\sum_{i=1}^{n}\left([y_i \mathit{log}(\mathit{sig}(Z_i)) + (1-y_i)\mathit{log}(1-\mathit{sig}(Z_i))] + p_{i1} + p_{i2} + p_{i3}\right). \quad (3.24)$$

The cross-entropy loss encourages the model to produce accurate document-level sentiment prediction.

3.4. Application

3.4.1. Data preprocessing and training scheme

AMIC 2.0 is applied to the Sentiment140 dataset ([18]) which consists of 1.6 million tweets with brief messages each limited to a maximum of 140 characters. Exploratory data analysis revealed that, on average, an individual tweet contains approximately 14 words or 68 characters. Manually labeling such a large dataset would be impractical due

to its size. To overcome this challenge, [18] adopted a technique introduced in [40], where emoticons were utilized as sentiment labels.

Emoticons are textual representations of facial expressions or emotions. For example, ":)" represents a smiley face, indicating positive sentiment, while ":(" represents a sad face, conveying negative sentiment. From the Twitter platform, [18] collected a dataset comprising 1.6 million tweets that contained positive or negative emoticons and used them as noisy labels for supervised training ([18]). These tweets collected were posted between April 6 and June 25 in 2009. In the dataset, emoticons were removed from the tweets to allow the model to focus on the textual information.

Out of the 1.6 million tweets, 800,000 were associated with negative sentiment and the other 800,000 with positive sentiment. Following the standard NLP preprocessing practice, we removed numbers and punctuations from the tweets and converted all words to lowercase. Furthermore, all emojis and user tags were removed from the tweets.

In contrast to AMIC, which employs Glove-300-Wiki pre-trained embeddings, AMIC 2.0 utilizes word2vec ([32]) to generate word embeddings using the current Sentiment140 dataset during data preprosessing. It is because Glove-300-Wiki was trained using the Wikipedia data, which mainly consists of documents written in formal language using proper words. The language in the wine review dataset is also standard, which makes it appropriate to use the Glove-300-Wiki embeddings in SA. However, the tweets in Sentiment140 were often written in informal language, so employing word2vec to generate wording embeddings allows AMIC 2.0 to use better representation of words in Sentiment140.

Another concern is that the construction of z_{ij} in Equation (3.21) introduces a potential identifiability issue. Note that z_{ij} is obtained as the product of four components, where v_{ij} and δ_{ij}^{l} both can take positive values and negative values. The identifiability prob-

lem occurs when the product of v_{ij} and δ^l_{ij} stays the same when their signs are flipped simultaneously.

Let's consider a scenario where word j in document i has a positive context-independent sentiment as well as a positive context-dependent sentiment. In such a case, the training process would encourage z_{ij} to take a positive value. Intuitively, this would mean that both v_{ij} and δ^l_{ij} should also be positive, as the context-independent sentiment remains positive and should not be flipped by the context. However, the identifiability issue arises because z_{ij} can remain positive with both v_{ij} and δ^l_{ij} taking a negative value.

To address this challenge and ensure that v_{ij} would take the correct sign, we propose a two-pass training scheme for AMIC 2.0. In the first pass of the training scheme, the algorithm only updates v_{ij}, which is trained using a modified approach. Instead of using the original equation $z_{ij} = v_{ij} \times \delta^s_{ij} \times \delta^g_{ij} \times \delta^l_{ij}$ in Equation (3.21), we modify it to be $z_{ij} = v_{ij}$ to only compute context-independent sentiment without cnsidering the influence of context. This modification helps to ensure that v_{ij} generates meaningful sentiment scores, where a positive value of v_{ij} corresponds to a positive sentiment and a negative value a negative sentiment. By isolating the training of v_{ij} in the first pass, AMIC 2.0 can produce more stable and accurate estimation of its value.

In the second pass of the training scheme, we reintroduce the other components δ^s_{ij}, δ^g_{ij}, and δ^l_{ij} in the calculation of z_{ij}. However, during this pass, v_{ij} remains fixed while we only update the other components in Equation (3.21). This step allows the model to effectively capture the contextual dependencies and respective sentiment shifts in the document. More importantly, it also ensures that a negative value of δ^s_{ij} indicates a flipped sentiment and a positive value a nonflipped sentiment between the context-independent sentiment and context-dependent sentiment of word j.

Regarding the tuning parameters for the penalties, c_2 is set to 1e-3, and both c_1 and c_3 are set to 1e-4, the same choice used in AMIC. The capped relative position q is set to 3,

which means the model does not consider relative positions beyond three words from the target words. The training scheme is depicted in Algorithm 2.

Algorithm 2: AMIC 2.0 training procedure

Data: X_i, y_i
Initialization:
First Pass (with modified z_{ij}):
while *not stop condition* **do**
    Calculate v_{ij} ;
    Evaluate the objective function with modified z_{ij};
    Update parameter b_v through gradient-descent
    if *stop condition reached* **then**
        break and got to Second Pass;
    else
        continue;
    end
end
Second Pass (with original z_{ij});
set q to 3;
set c2 to 1e-3, c1 and c3 to 1e-4;
while *not stop condition* **do**
    Calculate δ^g_{ij}, δ^l_{ij} , δ^s_{ij};
    Evaluate the objective function with original z_{ij};
    Update the rest of the model parameters through gradient-descent;
    if *stop condition reached* **then**
        break and got to Second Pass;
    else
        continue;
    end
end

3.4.2. Model performance comparison

To assess the predictive capability of AMIC 2.0, we have implemented a training-validation-test split on the Sentiment140 dataset. The dataset was divided into three parts using a 9:0.5:0.5 ratio. This means that 1.44 million tweets were used for training, while the remaining 160,000 tweets were evenly divided between the validation set and the test set. This splitting strategy allows us to train the model on a large amount of data, validate its performance on a separate set, and finally evaluate its generalization on an independent test set, where the latter two sets still have a decent sample size.

Through the use of the training-validation-test scheme and parameter selection based on the validation set, we can effectively evaluate the model performance in terms of its capability to make accurate predictions. To gain an objective assessment on the performance for AMIC 2.0, we have included a number of statistical and deep-learning models in the comparison. The results are presented in Table 3.1.

Table 3.1: Comparison of Model Performance on the Sentiment140 Dataset

Models	Predictive Accuracy (%)		Models	Predictive Accuracy (%)
Logistic Regression	78.40		AMIC 2.0	82.63
SVM (linear kernel)	77.89		AMIC 2.0 (using the algorithm in [47])	82.50
Naive Bayes	77.45		AMIC 2.0 (without local sentiment shifter)	81.32
Random Forest	70.61		AMIC	81.27
CNN	79.33			
BiLSTM	80.21			
BERT	86.72			

Among the statistical and machine learning models, logistic regression has the best performance with a classification accuracy of 78.40%. It is closely followed by the support vector machine with a linear kernel with a classification accuracy of 77.89%. Naive Bayes produces an accuracy of 77.45%, while Random Forest has the lowest performance with an accuracy of 70.61%. Among the deep-learning models, BERT outperforms all the others with the highest classification accuracy of 86.72%. The CNN and BiLSTM models yield classification accuracy of 79.33% and 80.21%, respectively.

AMIC 2.0 produces a classification accuracy of 82.63%. The version of AMIC 2.0 using the original relative positional representation algorhim in [47] performs similarly with an accuracy of 82.50%. AMIC 2.0 without the local sentiment shifter has a lower accuracy of 81.32%, which is comparable to AMIC's performance at 81.27%.

The little difference in the performance between AMIC 2.0 and AMIC 2.0 with the algorithm in [47] shows that our proposed algorithm which removes the Value vector from the self-attention mechanism does not compromise its performance. Notably, by removing the Value vectors for the calculation of the local sentiment shifter, our algorithm yields a more efficient model that requires roughly 30% fewer parameters for the calculation of the local sentiment shifter compared to the algorithm in [47]. On the other hand, the noticeable difference in the performance between AMIC 2.0 and AMIC 2.0 without the local sentiment shifter demonstrates the utility of incorporating positional information. The local sentiment shifter plays a crucial role in capturing local dependencies and generating more accurate context-dependent sentiment scores.

In general, the statistical models, such as logistic regression, support vector machine with a linear kernel, and Naive Bayes, exhibit relatively lower performance compared to the other models. The primary reason for this performance gap lies in their limited ability to capture context-dependent information present in the text, which is essential for accurate understanding of the sentiment conveyed in text.

In comparison, models like CNN and BiLSTM have shown better performance than the statistical models. This improvement can be attributed to their ability to process and learn from the contextual information in the input text. CNN is proficient at detecting local patterns, while BiLSTM can capture sequential dependencies in text, both contributing to their contextual awareness.

Despite their improved performance, both CNN and BiLSTM are outperformed by the family of AMIC models. The AMIC models leverage the self-attention mechanism, enabling them to effectively capture global dependencies across the entire input sequence. This ability to understand the relationship of words and phrases in text allows the AMIC models to gain a deeper understanding of the overall context and sentiment expressed, resulting in further improved performance.

BERT has outperformed all the other models, including the AMIC models. This result can be attributed to several factors. First, BERT is a much larger model, containing over 300 times the number of parameters compared to AMIC 2.0. This larger capacity enables BERT to learn more complex patterns and representations, contributing to its superior performance. Second, while all the other models are trained only on the Sentiment140 dataset, BERT stands out as the only pre-trained model, where its parameters are pre-trained based on a vast amount of various text data. In other wards, BERT, before it is applied to the Sentiment140 dataset, it has already gained superior capability as a large language model. When it is applied to the Sentiment140 dataset, it uses transfer learning, which is a machine learning technique where a model is first pre-trained on a large dataset and then fine-tuned on a specific task, enabling it to leverage knowledge from the pre-training phase for better performance on the current task. This pre-training process equips BERT with a deeper understanding of context and use of language, a characteristic that AMIC 2.0 lacks since it is trained exclusively with sentiment labels on the Sentiment140 dataset.

For simpler tasks such as wine reviews or medical report, the language used is usually more standard and domain-specific. In such cases, pre-training on a general language corpus might still offer some benefits, but the improvement in performance may not be as significant as in complex language tasks. This might explain why in the Wine Spectator dataset, BERT and AMIC's performances are similar.

On the other hand, complex tasks such as sentiment analysis of social media datasets like Sentiment140 involve highly diverse and informal language, including slang, emojis, and colloquial expressions. Pre-training on a vast amount of general language data, as what BERT does, allows the model to learn a rich understanding of language and context. This knowledge can be transferred to the downstream SA tasks, enabling the model to handle the complexities and nuances of sentiment expression in social media data more effectively.

Taken together the above considerations, it is expected that BERT will outperform AMIC 2.0 due to the advantages it holds, such as its larger size and the pre-training on a vast amount of data. However, despite BERT's dominance, AMIC 2.0 still demonstrates strong performance compared to the other models considered. This highlights the effectiveness of AMIC 2.0's approach and its ability to provide competitive results in SA.

Additionally, a significant distinction between BERT and AMIC 2.0 lies in the interpretability of their outputs. While BERT can produce word-level contextualized representations, these representations are not directly tied to the sentiment of words. This lack of direct association makes it challenging to interpret BERT's SA results from a clear decision-making perspective. On the other hand, AMIC 2.0 offers a higher level of interpretability. It explicitly identifies sentiment words versus function words, generates both context-independent and context-dependent sentiment scores for sentiment words, and employs two sentiment shifters to link these scores. Additionally, AMIC 2.0 calculates the document-level sentiment score by averaging word-level scores, providing clear explanation on how document-level sentiments are determined by the model. This transparency in the decision-making process makes AMIC 2.0 a valuable tool for SA, enabling researchers to gain a deeper understanding of the model's inner workings.

Finally, it is important to emphasize that the main focus of this study is to come up with an SA model that can provide interpretable results to help researchers to understand the decision-making process, while achieving the highest predictive performance is not the primary objective. AMIC 2.0 has demonstrated solid classification performance when compared to other popular SA models, with BERT being the only model to outperform it. The emphasis on interpretability and transparency distinguishes AMIC 2.0 as a valuable SA tool in domains where understanding the reasoning behind predictions is crucial.

3.4.3. Case study

With the integration of positional information and the utilization of local dependencies, AMIC 2.0 surpasses its predecessor, AMIC, in terms of its enhanced capability for nuanced and fine-grained SA. By examining AMIC 2.0's assessment of Sentence I and Sentence II and a number of other examples, we can illustrate AMIC 2.0's proficiency in accurately identifying negation and providing precise sentiment estimation based on word positional information.

Let us start with Sentence I (see Table 3.2). Sentence I consists of two clauses connected by a comma. In the first clause, the subject is "the service of the restaurant," which is modified by the adjective "good," indicating that the service is satisfactory. In the second clause, the subject is "the overall experience," which is modified by the phrase "not bad." Although "bad" itself carries negative sentiment, it is negated by "not". Putting it together, the sentence delivers a postive sentiment.

Table 3.2: AMIC 2.0's Analysis Result of Sentence I

Raw text	The service of the restaurant is good, the overall experience is not bad.												
Input text	the	service	of	the	restaurant	is	good	the	overall	experience	is	not	bad
v_{ij}	7.10	-1.0	6.9	7.1	19.3	-2.1	21.2	7.1	30.1	11.4	-2.1	-17.4	-28.1
δ_{ij}^{s}	0	0	0	0	0	0	1	0	0	0	0	1	1
δ_{ij}^{g}	-	-	-	-	-	-	1.47	-	-	-	-	6.6	2.6
δ_{ij}^{l}	-	-	-	-	-	-	0.45	-	-	-	-	-0.9	-0.8
Z_{ij}	0	0	0	0	0	0	14.2	0	0	0	0	103.8	62.5
Z_i	60.17												
Sentiment Label	Positive												

Table 3.2 provides a summary of the components used in the calculation of the context-dependent sentiment of individual words and the document-level sentiment label for Sentence I. The v_{ij} column contains the context-independent sentiment score, which remains constant regardless of the context. For instance, the context-independent sentiment of "good" is 21.2 in sentences it appears, likewise, the context-independent sentiment score of "bad" is always -28.1. The δ_{ij}^{s} column contains the indicator on whether a word is identified as a sentiment word. In Sentence I, AMIC 2.0 identifies "good," "not," and "bad" as

sentiment words. The δ_{ij}^{g} column reports the value for the global sentiment shifter, which adjusts the sentiment expressed based on all the other words in the text. The δ_{ij}^{l} column shows the estimates for the local sentiment shifter, which modifies the sentiment based on the adjacent words. The z_{ij} column represents the context-dependent sentiment score of each word. Finally, Z_i denotes the document-level sentiment score, calculated as the average sentiment of the sentiment words in the document.

We can see that AMIC 2.0 effectively handles negation in Sentence I by recognizing "bad" in the sentence is part of "not bad". The negative context-independent sentiment of "bad", -28.1, after being negated by "not", has a negative local shifter, -0.8, resulting in a positive context-dependent sentiment of 62.5.

Sentence II, like Sentence I, also consists of two clauses (see Table 3.3). The subjects, verbs, and grammatical structure of both clauses remain unchanged. The distinction lies in the adjectives used. In the first clause, the adjective "not good" introduces a negation and conveys negative sentiment. In the second clause, the adjective "bad" also conveys negative sentiment. Overall, the sentence expresses a sense of dissatisfaction towards both the service and the overall experience of the restaurant.

Table 3.3: AMIC 2.0's Analysis Result of Sentence II

Raw text	The service of the restaurant is not good, the overall experience is bad.												
Input text	the	service	of	the	restaurant	is	not	good	the	overall	experience	is	bad
v_{ij}	7.10	-1.0	6.9	7.1	19.3	-2.1	-17.4	21.2	7.1	30.1	11.4	-2.1	-28.1
δ_{ij}^{s}	0	0	0	0	0	0	1	1	0	0	0	0	1
δ_{ij}^{g}	-	-	-	-	-	-	6.59	1.46	-	-	-	-	2.66
δ_{ij}^{l}	-	-	-	-	-	-	0.79	-0.99	-	-	-	-	0.16
Z_{ij}	0	0	0	0	0	0	-91.7	-30.7	0	0	0	0	-11.7
Z_i	-44.37												
Sentiment Label	Negative												

As demonstrated in Table 3.3, AMIC 2.0 has identified three sentiment words in Sentence II: "not," "good," and "bad." In the first clause, the positive context-independent

sentiment score of “good” (21.2, the same value as in Sentence I) is reversed by “not” in the phrase “not good,” leading to a negative context-dependent sentiment score of -30.7. In the second clause, the sentiment of “bad” is not reversed because there are no negation words nearby, resulting in a context-dependent sentiment score of -11.7. The document-level sentiment score is -44.37, indicating an overall negative sentiment conveyed in Sentence II.

It is also interesting to point out the different treatment of “not” in these two sentences by AMIC 2.0. Note that ‘not” has the same content-independent sentiment score of -17.4 (a negative sentiment) in both cases. This makes sense because “not” is used to express negation, denial, refusal, or prohibition. In Sentence I, “not” is placed next to “bad”. Though both words carry a negative sentiment by themselves, together “not bad” conveys a positive sentiment. Thus, the sentiment polarity of “not” is flipped to a positive sentiment with 103.8 as its context-dependent sentiment score. In Sentence II, the sentiment of “not” is not flipped as it is positioned adjacent to “good”. It is even strengthened to have a more negative context-dependent sentiment score of -91.7, capturing the clearly negative sentiment expressed in the phrase “not good”. This comparison further demonstrates the capability AMIC 2.0 in providing nuanced understanding and accurate identification of sentiment in sentences that takes into account specific word positions.

In contrast, AMIC, which employs the orginal self-attention mechanism, does not incorporate positional information and thus ignores local dependencies. As a consequence, AMIC can not distinguish Sentence I and Sentence II because they have the same collection of words. So it produces the same result for the two sentences, which is included in Appendix B.2.

Note that natural language has a rich representation of negative expressions, where negation can be classified into different groups in different ways ([61]). For example, we can classify negation by the location of the negation word with respect to the negated concept, or we can make a distinction between negation in the asserted meaning (i.e.,

explicit negation) and negation in the non-asserted content (i.e., implicit negation). In this study, we focus on two groupings of negation: preceding negation vs. succeeding negation, and explicit negation vs. implicit negation.

In the first group, negation is classified based on the position of the negating word relative to the negated word. For example, NegEX ([5]), a simple algorithm for negation capturing, classifies negation triggers (i.e., negation words) into those that precede the negated concept and those that succeed the negated concept. Similarly, [35] also classifies negation words into preceding signals and succeeding signals. In this study, we refer to the type of negation where the negating word (such as "not") precedes the negated concept as preceding negation, where the succeeding negation is when the negation word is placed after the negated word.

In the second group, negation can be classified based on whether the negation cue is considered an explicit negation word or an implicit negation word ([8]). According to [8], explicit negation includes expressions such as "no," "not," "never.", etc. In contrast, implicit negation does not use these specific negation words. For example, words like "forget," "fail," "doubt," and "deny" are considered examples of implicit negation.

The negation structure in both Sentence I and Sentence II is considered to be preceding negation, because the negation word "not" precede the word whose sentiment is negated by it. It also falls into the category of explicit negation because of the use of explicit negation word "not". The following example tweet demonstrates the model's ability to capture succeeding negation with the word "free" (Table 3.4).

The model recognizes "celebrating," "phil," "cancer," and "free" as sentiment words, among which only "cancer" conveys a negative context-independent sentiment, while the other three words carry positive context-independent sentiment. In this case, the succeeding negation word "free" negates the sentiment conveyed by "cancer." Consequently, the context-independent sentiment score of "cancer" at -38.6 is shifted to a positive context-

Table 3.4: AMIC 2.0's Analysis Result of a Succeeding Negation Example

Raw text	Celebriting Phil being one year cancer free!						
Input text	celebriting	phil	being	one	year	cancer	free
v_{ij}	32.9	17.3	-11.5	2.3	-4.6	-38.6	19.9
δ_{ij}^{s}	1	1	0	0	0	1	1
δ_{ij}^{g}	1.98	1.63	-	-	-	1.4	2.03
δ_{ij}^{l}	0.73	0.79	-	-	-	-0.28	0.45
z_{ij}	47.80	22.5	0	0	0	14.9	18.5
Z_i	25.925						
Sentiment Label	Positive						

dependent sentiment score of 14.9, accurately interpreting the positive sentiment conveyed by the phrase "cancer-free." In contrast, the context-independent sentiment scores of "celebrating," "phil," and "free" remain unchanged, contributing to the overall positive sentiment conveyed in the tweet.

Next we present an example demonstrating AMIC 2.0's capability to automatically capture implicit negation. The tweet in Table 3.5 presents a sentence that contains implicit negation. In the main clause, "Chicago was awesome," a positive sentiment is expressed, portraying Chicago favorably. However, in the following clause, "my dreams were shattered" conveys a strong negative sentiment.

The model identifies "awesome," "although," "dreams," and "shattered" as sentiment words in the sentence. Note that the phrase "my dreams were shattered" contains implicit negation, as it conveys a negative sentiment without using explicit negation words like "not" or "never." The implicit negation is implied through the word "shattered". Because of it, the word "dreams" which carries a positive context-independent sentiment (11.3), is coupled with a negative local sentiment shifter, resulting in a negative context-dependent sentiment of "dreams" (-20.9), which correctly leads to a negative document-level sentiment. This example demonstrates that AMIC 2.0 can accurately capture implicit negation

Table 3.5: AMIC 2.0's Analysis Result of an Implicit Negation Example

Raw text	Chicago was awesome although my dreams were shattered.							
Input text	chicago	was	awesome	although	my	dreams	were	shattered
v_{ij}	1.7	-3.1	35.1	2.1	-7.4	11.3	-5.6	-25.5
δ^s_{ij}	0	0	1	1	0	1	0	1
δ^g_{ij}	-	-	1.30	0.96	-	4.25	-	1.31
δ^l_{ij}	-	-	0.04	0.87	-	-0.43	-	0.50
z_{ij}	0.0	0.0	1.7	1.7	0	-20.9	0.0	-16.8
Z_i	-8.57							
Sentiment Label	Negative							

relationships, showing the model's strength compared to rule-based negation systems that heavily rely on explicit negation words to identify negation.

In addition to negation handling, AMIC 2.0 can also deal with other types of language complexity, such as use of intensifiers, where intensifiers are adverbs or adverbial phrases that strengthen the meaning of other expressions and show emphasis. Our next example demonstrates how AMIC 2.0 can distinguish sentiment words under the influence of intensifiers. Table 3.6 displays two short phrases, "bad" and "very bad." Apparently, "very bad" delivers a stronger negative sentiment than just "bad". Note that the global sentiment shifter, without bearing positional awareness, can not recognize that "very" only intensifies the second "bad", and it takes the same value of "7.61" for both occurrences of "bad" in the sentence. However, the local sentiment shifter, equipped with positional awareness, accurately recognizes that "very" is a function word that only intensifies the sentiment of the second "bad" but not the first "bad". As a result, the local sentiment shifter for the second "bad" is larger that for the first "bad". resulting in a stronger negative sentiment for the second "bad" correctly.

In summary, the examples demonstrated from Table 3.2 to Table 3.6 exhibit a number of different representations of language complexity, covering preceding negation vs.

Table 3.6: AMIC 2.0's Analysis Result of an Intensifier Word

Raw text	Bad, very bad!		
Input text	bad	very	bad
v_{ij}	-28.1	3.8	-28.1
δ_{ij}^{s}	1	0	1
δ_{ij}^{g}	7.61	-	7.61
δ_{ij}^{l}	0.67	-	0.92
z_{ij}	-142.6	0.0	-196.7
Z_i	-169.7		
Sentiment Label	Negative		

succeeding negation, explicit negation vs. implicit negation, and presence of intensifier. All these examples have illustrated AMIC 2.0's capability of conducting fine-grained SA by incorporating word positional information, providing detailed interpretation of analysis process, and producing correct classification results in SA. It is worth noting that AMIC 2.0 achieves these goals without requiring an annotated dataset for training, which gives it a great advantage over the methods that need one.

3.4.4. Sentiment lexicon for Sentiment140

Similar to its predecessor AMIC, AMIC 2.0 can automatically generate a domain-specific sentiment lexicon. In this lexicon, each word has a sentiment score, which is the average of its context-dependent scores across all occurrences. Table 3.7 shows the top 50 words that have the highest sentiment scores in the Sentiment140 dataset, where the words can be put into several categories:

- Words that express joy and excitment such as "yayy," "rad," "yay," " woot" "congrats", "excited," and "stoked." These words are used to convey positive emotions, enthusiasm, and a sense of celebration.

Table 3.7: AMIC 2.0'S List of Top 50 Positive Sentiment Words

followfriday	congratulations	recommendation	welcome
appreciated	pleasure	smiling	lovin
loving	goodmorning	proud	compliment
ff	woot	congrats	coolest
sharing	rb	mothers	feedback
cheers	greetings	props	entertaining
smile	blessed	sweetest	thanks
myweakness	twitterverse	appreciate	add
thankyou	chillin	hilarious	excited
worries	yayy	ty	rad
hello	glad	yey	deserved
yaay	yumm	thanx	sir
inspiring	stoked		

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

- Words that convey a feeling of gratitude and appreciation such as "acknowledged," "grateful," "thanks," "gratitude," and "gratefulness." These expressions are frequently utilized to recognize and convey appreciation for something.
- Words that evoke feelings of happiness and contentment usch as "pleasure," "smiling," "loving," "glad," and "blessed". These words reflect a positive outlook and a sense of well-being.
- Words that describe positive experiences or positive aspects such as "recommendation," "goodmorning," "proud," "sharing," and "entertaining." These words highlight enjoyable or favorable situations.

Five words"followfriday," "ff," "rb," "add," and "sir" in the top 50 list seem to be less obvious to carry positive sentiment. Out of context, these words may not inherently carry

positive sentiment. The sentiment associated with these words largely depends on the context and the conventional usage in tweets.

The hashtag or term “followfriday” is often used to recommend or highlight other Twitter users, expressing appreciation for their content or recommending them to one’s followers. Similarly, “ff” is an abbreviation for “Follow Friday.” When users include the “ff” hashtag along with the usernames of others, it indicates that they are recommending those accounts to their followers. It is a way to show appreciation and promote other users, creating a positive and supportive atmosphere.“rb” stands for “Retweet” or “Reblog.” When someone retweets or reblogs a post, it indicates that they find the content worth sharing with their own followers. This act of sharing and highlighting someone else’s content is often seen as positive and supportive. In the context of tweets, “add” may refer to the action of adding someone to a list, group, or conversation. When someone asks to be added or offers to add others, it can create a sense of inclusion and connection, fostering a positive community or networking environment. While “sir” is a term to show respect, its sentiment in tweets can vary depending on the context and the relationship between the users. Majority of the tweets that uses the word in a positive way to show politeness, deference, or admiration. Overall, the positive sentiment associated with these words in tweets is often derived from the social dynamics, supportive nature, and collaborative atmosphere in the context of Twitter interactions.

Table 3.8 presents the bottom 50 words with the lowest sentiment scores in this study, which can be roughly categorized into the following groups:

- Words that convey feelings of pain such as “ache,” “cramps,” “toothache,” “swollen,” “headache,” “itchy,” “hurts,” “infection,” “bleeding,” “hayfever,” and “stomach.” These words evoke physical discomfort and highlight the experience of pain.
- Words that evoke a sense of sadness and melancholy such as “sad,” “miserable,” “gutted,” “crying,” “lonely,” “heartbroken,” “disappointed,” “gloomy,” “depressing,” and

Table 3.8: AMIC 2.0'S List of 50 Words with Lowest Sentiment

fawcett	farrah	rip	fathers	ache
carradine	died	miserable	gutted	cramps
toothache	itchy	swollen	headache	icky
sad	misses	ruined	poorly	crying
missin	tummy	lonely	unfair	disappointing
tragic	hurts	canceled	sinus	sadly
heartbroken	disappointed	hayfever	oww	bummed
gloomy	depressing	throat	cancelled	ouch
infection	depressed	stressed	hates	worst
burnt	worried	bleeding	stomach	dies

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

"depressed." They depict feelings of sorrow, disappointment, and a sense of emotional heaviness.

- Words that express disappointment, signaling unmet expectations or unfavorable outcomes such as "ruined," "poorly," "missin," "unfair," "disappointing," and "cancelled." These words reflect a sense of letdown and dissatisfaction.

Among these 50 words with the lowest sentiment, three words stand out as least expected: "fawcett," "farrah," and "fathers." Upon closer examination, we find out that many tweets with clear negative sentiment were posted at the time when the movie actress Farrah Fawcett passed away (on June 25th, 2009). As a result, these tweets expressed feelings of sadness and mourning for the loss of the star, leading to an overall negative sentiment score associated with the term.

Notice that "farrah" has less negative sentiment associated with it than "fawcett." It is because "fawcett" is an unusual first name, exclusively used in tweets referring to Farrah

Fawcett, while “farrah” is a more common name and has been used in tweets unrelated to the mentioning of the movie star. For instance, there are tweets expressing excitement about a road trip with someone named Farrah, such as “congratulations best wishes amp lots of happiness auntie farrah.” This difference explains why “farrah” has less negative sentiment than “fawcett” assigned by AMIC 2.0.

The word “fathers” is frequently used in tweets related to Father’s Day, which, in a strange way, tends to carry negative connotations. Examples include sentiments like “I hate Father’s Day now; I have to do dad’s jobs today” or “I still don’t have an idea for a Father’s Day gift. It’s getting closer to gift time. What do I do?” These tweets reflect a predominantly negative sentiment associated with Father’s Day, possibly stemming from stress caused by the holiday.

The application to Sentiment 140 in this section demonstrates AMIC 2.0’s capability to create a domain-specific sentiment lexicon. The domain-specific sentiment lexicon enables us to assess the sentiments of documents more accurately in the domain of interest, provide more meaningful interpretation, and gain a deeper understanding of the domain-specific wording characteristics.

3.5. Discussion

In Chapter 2, we introduced AMIC for word-level context-based sentiment analysis. AMIC can provide interpretable WCSA while keeping comparable performance to the state-of-the-art deep learning models. However, a limitation of AMIC is its ignorance of positional information of input words. As a result, its ability to accurately measure word-level and document-level sentiment is affected, especially in domains where sentiment heavily relies on the precise positions of words, as illustrated in the examples of Sentence I and Sentence II.

We have two goals in the construction of AMIC 2.0 to bring improvement over AMIC. The first goal is to incorporate positional information in WCSA. This improvement allows the model to effectively utilize positional information of words, enabling it to capture local dependencies and achieve more precise sentiment estimation. The second goal is to develop a model structure that explicitly demonstrates how context influences sentiment. This feature will help people to better understand the interaction of words and provide more detailed interpretation of the inner workings of the model.

To achieve the first goal, we have proposed an algorithm to incorporate relative positional embedding in the self-attention algorithm, so that the attention adjusted embeddings also carry positional information. Equipped with positional understanding, AMIC 2.0 has an enhanced ability to capture the intricate nuances of sentiment analysis.

One notable advantage of the proposed algorithm is its efficiency. Compared to the approach proposed in [47] to incorporate positional information, AMIC 2.0 requires around 30% fewer parameters while maintaining its performance. This reduction in parameters not only streamlines the computational requirements but also contributes to improved training and inference speed.

As demonstrated in Section 3.4.3, AMIC 2.0 can effectively deal with nuanced sentiment phenomena, such as the impact of negation. By considering the positional information of words in relation to negation markers and excluding the value representation which concerns words' semantic meaning, the model becomes adept at accurately discerning the shift in sentiment caused by negation. This heightened sensitivity to negation greatly improves the model's overall sentiment analysis capability, allowing for more accurate and nuanced understanding of contextual sentiment.

To achieve the second goal, we have modified the model structure of AMIC to decompose the single contextualized word-level sentiment score into two scores: a context-independent sentiment score and a context-dependent sentiment score. The context-

independent sentiment score represents the inherent sentiment associated with individual words, regardless of the context. The context-dependent sentiment score modifies context-independent score by considering the context of the word, providing a more accurate measurement of sentiment in the particular contextual environment.

Furthermore, we have introduced a global sentiment shifter and a local sentiment shifter, with the former focusing on the overall interaction of words across the entire text and the latter on the immediate relationships between neighboring words. The two sentiment shifters together can provide a more detailed explanation on the inner workings of the model, showing how the context-independent sentiment of a word is influenced by both global and local dependencies which transforms it into the context-dependent sentiment. This dual-scoring approach, together with the two sentiment shifters, offers valuable insights into the model's decision-making process.

It is worth mentioning that AMIC 2.0 operates without requesting an annotated dataset which contains, for example, annotated negation words. Creating such datasets can be a time-consuming, labor-intensive, and error-prone task. Thus, a model that automatically handles language complexity such as different types of negation can significantly reduce the reliance on annotated datasets, resulting in greater efficiency and cost-effectiveness in sentiment analysis.

Moving forward, we plan to apply AMIC 2.0 to a broad spectrum of datasets across various domains. The investigation can provide us with a more comprehensive understanding of the model's performance and interpretability when it is applied to diverse datasets encompassing different language use complexities.

The second potential direction we are considering addresses a critical limitation of neural networks. In addition to being criticized as black-box models, neural networks have also faced scrutiny for their inability to provide uncertainty analysis, which can lead to overly confident predictions or the underestimation of potential risks.

One of the key strengths of our model lies in its shallow and relatively compact structure. This characteristic presents an opportunity for us to explore the integration of uncertainty measurement using Bayesian neural network approaches. By incorporating Bayesian neural networks into AMIC 2.0, we can introduce uncertainty estimation as an integral aspect of the model's predictive capabilities. Unlike traditional neural networks that treat model parameters as fixed values, Bayesian neural networks represent these parameters as probability distributions. This approach enables us to quantify uncertainty in predictions, leading to more reliable and well-calibrated results.

APPENDIX A

APPENDIX of CHAPTER 2

A.1. The Results on the Multiple Runs for the Comparison Study

Table A.1 displays the results from multiple runs of the SA models in the comparison study in Section 2.4.2. The table reports the average predictive accuracy and its standard deviation across the five runs following the same 18:1:1 training-validation-test ratio. The conclusion from this study on model performance is consistent with that in Section 2.4.2. Specifically, AMIC and BERT exhibit the best performances, followed by CNN and BiLSTM. The two MNIR models yield lower performance.

Table A.1: Comparison of Model Performances based on Multiple Runs

	Model	Mean Predictive Accuracy (Standard Deviation)
Word-level SA models	AMIC	0.8918 (0.00368)
	MNIR	0.8479 (0.00229)
	MNIR (with bigram)	0.8375 (0.00155)
Neural network Models	CNN	0.8802 (0.00327)
	BILSTM	0.8865 (0.00197)
	BERT	0.8912 (0.00379)

A.2. The Sentiment Word Lists for AMIC and MNIR

This appendix provides the lists of the top 100 positive sentiment words and the bottom 100 negative sentiment words from AMIC and MNIR, respectively. AMIC's unique words are marked in red, MNIR's unique words are marked in blue, overlapping words are marked in the black.

Table A.2

AMIC's List of Top 100 Positive Sentiment Words

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
excellent	rain	wears	artfully	neatly
beaded	remarkably	clad	effortlessly	bouquet
aftertaste	beguiling	luxurious	rolls	foie
tempting	terroir	deceptively	luscious	jujube
baby	brownie	buckwheat	strudel	barolo
gras	glowing	brilliant	tongue	fine
classy	effortless	richly	masculine	delicacy
nimble	blancs	cuttings	embers	pecan
corn	violets	alluring	drape	drapes
crafted	lush	melts	expressive	toffee

MNIR's List of Top 100 Positive Sentiment Words

stunning	superb	drc	impeccable	incredibly
endless	upon	chave	incredible	stunner
luxuriant	wellsubmerged	torrent	supersilky	strikingly
leroy	cascading	lalande	pichon	maximum
authoritatively	glorious	gloriously	eiswein	gracefulness
extraordinary	hedonists	auction	winner	doors
tremendously	soar	knockout	saturating	aftertastenonblind
lengthnonblind	santo	multidimensional	dovetailing	marshaling
inner	anisefilled	unimpeded	gorgeous	tremendous
amazingly	beautiful	beauty	february	amazing
cascade	boding	waves	civilized	gorgeously
pieces	remarkable	thoroughly	intricate	beautifully
ethereal	lardière	january	superlong	drips
evolution	riveting	captivating	superracy	profound
potential	gripfilled	impeccably	honed	waiting
remarkably	brunello	pinpoint	impenetrable	kaleidoscope
pulses	cruises	seamless	brilliant	explosive
innate	sgn	block	rising	multilayered
gushes	wealth	pastissoaked	large	resist
adelaide	yattarna	largescaled	tba	november

Table A.3

AMIC's List of Bottom 100 Negative Sentiment Words

quick	generic	hearted	simple	canned
uncomplicated	diluted	tinny	neutral	straightforward
stale	cocktail	easygoing	fizzy	picnic
flat	lovage	greenish	unfocused	breezy
beaujolais	metallic	dull	easy	tail
modestly	decent	fade	scallion	modest
cucumber	cloying	watermelon	soft	parsley
asparagus	kosher	muddled	herbal	lemonade
detract	weedy	blunt	tired	muscadet
grass	grassy	chilled	trim	overripe
alcoholic	drying	muted	onion	crab
loosely	grape	stewed	astringent	coarse
drinking	cooked	clipped	pith	thin
maraschino	flabby	effervescent	short	odd
herbaceous	spritz	sour	veneer	washed
pea	offbeat	musty	cider	soapy
rustic	slightly	bottles	softly	tinned
albeit	tails	colombard	faint	pulp
iced	plodding	leathery	kilter	acidic
woody	prosecco	jerky	lemongrass	bitterness

MNIR's List of Bottom 100 Negative Sentiment Words

quick	hoping	canned	cava	stewy
hardcandy	prosecco	quaffer	quaffable	tinned
flat	userfriendly	unfocused	bing	lambrusco
overwhelm	vaz	tuttifrutti	saint	neutral
antão	aragonês	colombard	spritzy	mulchy
roupeiro	limited	canelli	uncomplicated	murky
overshadows	lemberger	wellchilled	fina	thins
roditis	crab	picnic	generic	underripe
fizzy	soapy	torrontés	pleaser	xarello
catarratto	regional	offbeat	barnyardy	tibouren
issue	weedy	threaten	tinny	crémant
unbalanced	tooth	moschofilero	nouveau	lumbering
stale	seashell	schiava	gussied	greenish
lemonade	crispedged	lighthearted	tastings	brachetto
detract	diluted	lovage	easyenjoy	burgers
washed	unoaked	vidal	grapefruitflavored	limeade
sauvignonlike	simple	basic	blowsy	simpler
sipper	straightforward	dilute	spritz	tailing
decent	hollow	gamay	cocktail	easydrinking
tails	castelão	struggles	resiny	tired

APPENDIX B

APPENDIX of CHAPTER 3

B.1. The self-attention mechanism algorithm

In this appendix, we describe the implementation steps of the self-attention mechanism. There is a difference in notation between equations presented here and those in Section 3.2.1, where the equations in the appendix have been modified to include the subscript i, representing the ith document.

Let x_{ij} denote the d-dimensional word embedding of a word j in the ith document. Consider an input sequence of word embedding vectors for a document denoted by $X_i = [x_{i1}, x_{i2}, ..., x_{im}]$, where $x_{ij} \in \mathbb{R}^d$ represents the embedding for the j^{th} word in i^{th} document and m is the total number of words in the sequence. Self-attention uses three $d \times d$ projection matrices W^Q, W^K, and W^V, which are updated as model parameters during training. These matrices serve to project the inputs into Query, Key, and Value vectors for each word, respectively, via matrix multiplication between the matrices W and the embedded inputs x_{ij},

$$\text{Query vector} : x_{ij}^Q = x_{ij}W^Q$$

$$\text{Key vector} : x_{ij}^K = x_{ij}W^K$$

$$\text{Value vector} : x_{ij}^V = x_{ij}W^V.$$

Next the self-attention mechanism proceeds to compute a weight vector for each word, considering its interactions with other words in the sequence. For instance, in the self-attention representation of word j, the similarity between word j and word k in document i is computed as follows:

$$e_{ijk} = (x_{ij}^{Q})(x_{ik}^{K})^{T} \tag{B.1}$$

The attention weight α_{ijk} is then computed as the standardized similarity score of e_{ijk}:

$$\alpha_{ijk} = \frac{\textit{exp}(e_{ijk})}{\sum_{h=1}^{m} \textit{exp}(e_{ijh})}, \tag{B.2}$$

where α_{ijk} can be interpreted as the level of attention that should be given to the k^{th} word when calculating the context-dependent representation for the j^{th} word in document i.

In the third and final step, the self-attention mechanism computes a weighted sum of the Value vectors for each word in the input sequence. The weights, calculated in the second step, measure the relevance of each word in the input sequence to the current word. The self-attention embedding of word j, r_{ij}, which incorporates the context of word j, is computed as follows:

$$r_{ij} = \sum_{k=1}^{m} \alpha_{ijk}(x_{ik}^{V}). \tag{B.3}$$

The self-attention representation for the word j within document i is denoted as r_{ij}, which represents the result of applying self-attention transformation to the word j. Denote $R_i = [r_{i1}, r_{i2}, ..., r_{im}]$ as the transformed representation of document i. we use $F^{\textit{self-attention}}()$ as an aggregated function for all the above steps:

$$R_i = F^{\textit{self-attention}}(X_i), \tag{B.4}$$

where $X_i = [x_{i1}, x_{i2}, ..., x_{im}]$ consists of the original word embeddings in document i.

B.2. AMIC output of Sentence I and Sentence II

Table B.1 and B.2 display the outputs of AMIC for Sentence I and Sentence II, respectively. It is evident that AMIC assigns exactly the same word-level contextualized sentiment score to each word in both sentences, leading to an identical document-level sentiment prediction of -0.683 for both Sentence I and Sentence II. This resulted in an incorrect prediction of sentiment label for Sentence I.

Additionally, it is worth highlighting that the word “good” is assigned a negative contextualized sentiment score of -30.4 in both sentences, despite “good” typically being associated with a positive sentiment and its sentiment is not flipped in Sentence I.

By comparing Table B.1 and B.2 to Table 3.3 and 3.4, it becomes evident that AMIC 2.0’s positional awareness provides it with a clear advantage over AMIC. This positional awareness allows AMIC 2.0 to accurately assess the sentiment of individual words in different local contexts, resulting in more precise sentiment predictions.

Table B.1: AMIC’s Analysis Result of Sentence I

Raw text	The service of the restaurant is good, the overall experience is not bad.												
Input text	the	service	of	the	restaurant	is	good	the	overall	experience	is	not	bad
δ_{ij}	0	1	0	0	1	0	1	0	0	0	0	1	1
Z_{ij}	0	-16.8	0	0	-8.5	0	-30.4	0	0	0	0	39.3	23
Z_i	-0.683												
Sentiment Label	Negative												

Table B.2: AMIC’s Analysis Result of Sentence II

Raw text	The service of the restaurant is not good, the overall experience is bad.											
Input text	the	service	of	the	restaurant	not	good	the	overall	experience	is	bad
δ_{ij}	0	1	0	0	1	1	1	0	0	0	0	1
Z_{ij}	0	-16.8	0	0	-8.5	39.3	-30.4	0	0	0	0	23
Z_i	-0.683											
Sentiment Label	Negative											

BIBLIOGRAPHY

[1] Alfred V. AHO. Chapter 5 - algorithms for finding patterns in strings. In JAN VAN LEEUWEN, editor, *Algorithms and Complexity*, Handbook of Theoretical Computer Science, pages 255–300. Elsevier, Amsterdam, 1990.

[2] Ivan Bilan and Benjamin Roth. Position-aware self-attention with relative positional encodings for slot filling, 2018.

[3] Erik Cambria, Björn Schuller, Yunqing Xia, and Catherine Havasi. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, 28(2):15–21, 2013.

[4] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77:329–353, 2018.

[5] Wendy Chapman, Will Bridewell, Paul Hanbury, Gregory Cooper, and Bruce Buchanan. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of Biomedical Informatics*, 34:301–310, 11 2001.

[6] Kehai Chen, Rui Wang, Masao Utiyama, and Eiichiro Sumita. Context-aware positional representation for self-attention networks. *Neurocomputing*, 451:46–56, 2021.

[7] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas, November 2016. Association for Computational Linguistics.

[8] Herbert H. Clark. *Semantics and Comprehension*. The Hague: Mouton, 1976.

[9] Isaac Councill, Ryan Mcdonald, and Leonid Velikovich. What's great and what's not: Learning to classify the scope of negation for improved sentiment analysis. pages 51–59, 08 2010.

[10] Isaac Councill, Ryan McDonald, and Leonid Velikovich. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Workshop on Negation and Speculation in Natural Language Processing*, 2010.

[11] Noa Cruz Diaz, Maite Taboada, and Ruslan Mitkov. A machine learning approach to negation and speculation detection for sentiment analysis. *Journal of the American Society for Information Science and Technology (JASIST)*, 06 2015.

[12] Sanjiv R. Das and Mike Y. Chen. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375–1388, 2007.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[14] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1):31–71, 1997.

[15] Gerald A. Fisher and Manfred Weber. Lalr(1) parsing for languages without reserved words. *SIGPLAN Not.*, 14(11):26–30, nov 1979.

[16] Matthew Gentzkow, Bryan Kelly, and Matt Taddy. Text as data. *Journal of Economic Literature*, 57(3):535–74, September 2019.

[17] Hamidreza Ghader and Christof Monz. What does attention in neural machine translation pay attention to?, 2017.

[18] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6, 2009.

[19] Gregory Grefenstette, Yan Qu, James Shanahan, and David Evans. Coupling niche browsers and affect analysis for an opinion mining application. pages 186–194, 01 2004.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[21] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

[22] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer, 2018.

[23] Tomoki Ito, Kota Tsubouchi, Hiroki Sakaji, Tatsuo Yamashita, and Kiyoshi Izumi. Contextual sentiment neural network for document sentiment analysis. *Data Science and Engineering*, 5, 06 2020.

[24] Lifeng Jia, Clement Yu, and Weiyi Meng. The effect of negation on sentiment analysis and retrieval effectiveness. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 1827–1830, New York, NY, USA, 2009. Association for Computing Machinery.

[25] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009.

[26] Duwani Katumullage, Chenyu Yang, Jackson Barth, and Jing Cao. Using neural network models for wine review classification. *Journal of Wine Economics*, 17(1):27–41, 2022.

[27] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[28] Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, May 2012.

[29] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[30] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113, 2014.

[31] Saeed Mehrabi, Anand Krishnan, Sunghwan Sohn, Alexandra M. Roch, Heidi Schmidt, Joe Kesterson, Chris Beesley, Paul Dexter, C. Max Schmidt, Hongfang Liu, and Mathew Palakal. Deepen: A negation detection system for clinical text incorporating dependency relation into negex. *Journal of Biomedical Informatics*, 54:213–219, 2015.

[32] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[33] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

[34] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[35] Pradeep Mutalik, Aniruddha Deshpande, and Prakash Nadkarni. Use of general-purpose negation detection to augment concept indexing of medical documents:

A quantitative study using the umls. *Journal of the American Medical Informatics Association : JAMIA*, 8:598–609, 11 2001.

[36] Joakim Nivre. Dependency grammar and dependency parsing. 2005.

[37] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics, July 2002.

[38] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[39] Soumya Ray and David Page. Multiple instance regression. In *International Conference on Machine Learning*, 2001.

[40] Jonathon Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL Student Research Workshop*, pages 43–48, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[41] Andrew J Reagan, Christopher M Danforth, Brian Tivnan, Jake Ryland Williams, and Peter Sheridan Dodds. Sentiment analysis methods for understanding large-scale texts: a case for using continuum-scored words and word shift graphs. *EPJ Data Sci.*, 6:28, 2017.

[42] Johan Reitan, Jørgen Faret, Björn Gambäck, and Lars Bungum. Negation scope detection for twitter sentiment analysis. pages 99–108, 01 2015.

[43] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019.

[44] Guergana Savova, James Masanz, Philip Ogren, Jiaping Zheng, Sunghwan Sohn, Karin Kipper-Schuler, and Christopher Chute. Mayo clinical text analysis and knowledge extraction system (ctakes): Architecture, component evaluation and applications. *Journal of the American Medical Informatics Association : JAMIA*, 17:507–13, 09 2010.

[45] Yeon Seonwoo, Ji-Hoon Kim, Jung-Woo Ha, and Alice Oh. Context-aware answer extraction in question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2418–2428, Online, November 2020. Association for Computational Linguistics.

[46] Erhan Sezerer and Selma Tekir. A survey on neural word embeddings. *ArXiv*, abs/2110.01804, 2021.

[47] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations, 2018.

[48] Prakash Kumar Singh and Sanchita Paul. Deep learning approach for negation handling in sentiment analysis. *IEEE Access*, 9:102579–102592, 2021.

[49] Sunghwan Sohn, Stephen T Wu, and Christopher G. Chute. Dependency parser-based negation detection in clinical narratives. *AMIA Summits on Translational Science Proceedings*, 2012:1 – 8, 2012.

[50] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[51] György Szarvas, Veronika Vincze, Richárd Farkas, and János Csirik. The BioScope corpus: annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 38–45, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[52] Sayyida Tabinda Kokab, Sohail Asghar, and Shehneela Naz. Transformer-based deep learning models for the sentiment analysis of social media data. *Array*, 14:100157, 2022.

[53] Matt Taddy. Multinomial inverse regression for text analysis. *Journal of the American Statistical Association*, 108(503):755–770, 2013.

[54] Matt Taddy. Distributed multinomial regression. *The Annals of Applied Statistics*, 9(3):1394 – 1414, 2015.

[55] Duyu Tang, Furu Wei, Bing Qin, M. Zhou, and Ting Liu. Building large-scale twitter-specific sentiment lexicon : A representation learning approach. In *International Conference on Computational Linguistics*, 2014.

[56] Abhilasha Tyagi and Naresh Sharma. Sentiment analysis using logistic regression and effective word score heuristic. *International Journal of Engineering and Technology(UAE)*, 7:20–23, 04 2018.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[58] Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C.-C. Jay Kuo. Evaluating word embedding models: methods and experimental results. *APSIPA Transactions on Signal and Information Processing*, 8(1):–, 2019.

[59] Le Wang and Rui Xia. Sentiment lexicon construction with representation learning based on hierarchical sentiment supervision. In *Conference on Empirical Methods in Natural Language Processing*, 2017.

[60] Peter Willett. The porter stemming algorithm: Then and now. *Program electronic library and information systems*, 40, 07 2006.

[61] Ming Xiang, Julian Grove, and Anastasia Giannakidou. Explicit and implicit negation, negative polarity, and levels of semantic representation. 03 2014.

[62] Danyi Xiong. Bayesian multiple instance learning with application to cancer detection using tcr repertoire sequencing data. *Statistical Science Dissertation at Southern Methodist University.*, 2022.

[63] Chenyu Yang, Jackson Barth, Duwani Katumullage, and Jing Cao. Wine review descriptors as quality predictors: Evidence from language processing techniques. *Journal of Wine Economics*, 17(1):64–80, 2022.

[64] Kiduk Yang. Widit in trec 2008 blog track: Leveraging multiple sources of opinion evidence. 01 2008.

[65] Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 73–78, Shanghai, China, October 2015.

[66] Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015.